Technische Universität Darmstadt
Fachgebiet Flugmechanik und Regelungstechnik



Studienarbeit im Studiengang Elektrotechnik
Fachrichtung Elektromechanische Konstruktionen

Auslegung und Realisierung der Dämpfungseigenschaften eines bestehenden Systems aktiver Sidesticks mit dem Ziel der Kopplung dieser Bedienelemente

vorgelegt von:
Daniel Prutz

Betreuer
Professor Dr. Ing. W. Kubbat
Professor Dr. Ing. H. Schlaak
Dipl. Ing. Jens Kaibel

4. Februar 2002

Zusammenfassung

Diese Arbeit schildert die Weiterentwicklung der aktiver Sidesticks, welche im Forschungssimulator des Institutes für Flugmechanik und Regelungstechnik zum Einsatz kommen.
Im Rahmen der Arbeit wurde die Problematik der Stickregelung neu betrachtet und dadurch wichtige Erkenntnisse über die Stabilität der Sticks im Einzelbetrieb, wie auch im gekoppelten Betrieb gewonnen.
Um ein instabiles Verhalten zu verhindern, wurde eine zusätzliche Dämpfung in das System integriert. Dazu wurden zusätzliche Inkrementalgeber in die Stickhardware eingebaut und deren Signale zur Berechnung eines Dämpfungssignal in die Sticksoftware implementiert.
Weiterhin wurde das existierende Konzept für eine Kopplung der beiden Sticks grundlegend überarbeitet und so eine stabile Kopplungsmöglichkeit der beiden Sidesticks realisiert.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit alleine und ausschließlich unter Verwendung der angegebenen Quellen erstellt habe.

Darmstadt, den 04.02.2002

Daniel Prutz

Inhaltsverzeichnis

1 Einführung ... 7

1.1 Der Forschungssimulator ... 7

1.2 Einbindung der Stickrechner im Forschungssimulator ... 8

1.3 Das Sticksystem ... 9

1.3.1 Mechanik und Motoren ... 9

1.3.2 Sensoren ... 10

1.3.3 Elektronik ... 10

1.3.4 Software ... 11

1.4 Die Aufgabe ... 12

2 Problemanalyse ... 14

2.1 Das Regelungstechnische Blockschaltbild ... 14

2.1.1 Erörterungen zum Blockschaltbild ... 15

2.2 Bestimmung der Übertragungsfunktion: ... 16

2.3 Analyse der zusätzlichen Dämpfung: ... 20

2.3.1 Bestimmung der neuen Übertragungsfunktion ... 20

2.4 Problemanalyse der Kopplung ... 22

2.4.1 Kopplungsprinzipien ... 23

2.4.2 Regelkreislauf der Kopplung ... 25

3 Inkrementalgeber Hardware ... 27

3.1 Konzept Bildung ... 27

3.1.1 Prinzipien ... 27

3.1.2 Lösungsvarianten ... 30

3.2 Realisierung der Inkrementalgeber Mechanik ... 35

4 Die Software ... 39

4.1 Programmierung der Inkrementalgeberkarte ... 39

4.2 Die Sticksoftware ... 43

4.3 Erweiterung der Sticksoftware (Konzept) ... 44

4.3.1 Implementierung des Inkrementalgebers ... 44

4.3.2 Implementierung der Kopplung ... 45

4.4 Realisierung des Softwarekonzepts ... 45
4.4.1 Veränderungen zur Implementierung der Inkrementalgebersignale ... 45
4.4.2 Veränderung der Sticksoftware für die Kopplung ... 46
5 Kritik und Ausblick ... 48
6 Literaturverzeichnis ... 49

Anhang

I Zusammenfassung Regelungstechnik ... 2
I.1 Legende zum Blockschaltbild (mit Dämpfungstabelle) ... 3
I.2 Berechnungen zum Regelungskreislauf ... 5
I.3 Simulink Ausdrucke ... 7
II Testergebnisse ... 9
II.1 Ausdrucke der Excel Graphen ... 9
III Mechanik ... 11
III.1 Berechnungen zur Inkrementalgebermechanik ... 11
III.2 Stückliste ... 12
III.3 Technische Zeichnungen ... 13
III.4 Umbau Stick Checkliste ... 21
IV Der Umgang mit dem Sticksystem ... 22
IV.1 Ausführbare Dateien ... 22
IV.2 Konfigerationsdatei Stick.cfg ... 23
IV.3 Einstellbare Dämpfungswerte ... 25
IV.4 Speicherort des Quellcodes auf der Festplatte ... 26
IV.5 Wichtige Programmteile der Sticksoftware ... 29
IV.6 Programmteile der Inkrementalgeber ... 29
IV.7 Kompilieren des Codes in .lib / .exe ... 30
IV.8 Unterschiede der Sticksysteme ... 31
V Kurzbeschreibung der Erweiterungen ... 32
VI Inkrementalgeber Programmierung ... 35

VI.1 Testsoftware Beschreibung 35
VI.2 Flussdiagramme zur Testsoftware 38

VII Sticksoftware Veränderungen im Detail 41

VII.1 Veränderungen zur Implementierung der Inkrementalgeber Software 41
VII.2 Veränderungen der Sticksoftware zur Kopplung der Sticks 46

VIII Softwarecode Testprogramme 49

IX Softwarecode Inkrementalgeber Implementierung 57

Abbildungsverzeichnis

Abbildung 1.1 : Der Forschungssimulator....9
Abbildung 1.2 : Einbindung des Sticksystems.... 9
Abbildung 1.3 : Die Stickmechanik....9

Abbildung 2.1 : Regelungstechnische Blockschaltbild (Grundsystem)....14
Abbildung 2.2 : Blockschaltbild vereinfacht (ohne A/D Wandler)....17
Abbildung 2.3 : Blockschaltbild vereinfacht (mit A/D Wandler)....18
Abbildung 2.4 : Regelungstechnische Blockschaltbild (mit Inkrementalgeber).... 20
Abbildung 2.5 : Blockschaltbild vereinfacht (mit Inkrementalgebern)....21
Abbildung 2.6 : mechanische Kopplung....22
Abbildung 2.7 : elektromechanische Kopplung....22
Abbildung 2.8 : einfache Winkelkopplung....23
Abbildung 2.9 : doppelte Winkelkopplung....23
Abbildung 2.10 : doppelte Drehmomentkopplung.... 24
Abbildung 2.11 : zwei wirkende Handkräfte....24
Abbildung 2.12 : Regelungstechnische Blockschaltbild der Kopplung.... 25

Abbildung 3.1 : Platzangebot im Stickgehäuse....35
Abbildung 3.2 : WellenkupplungAbbildung 3.3 : Motor mit Kupplung....35
Abbildung 3.3 : Motor mit Kupplung....36
Abbildung 3.4 : Inkrementalgebermechanik....36
Abbildung 3.5 : Inkrementalgeber Mechanik am Rollmotor (Copilotstickgehäuse)....37
Abbildung 3.6 : Inkrementalgeber Mechanik am Nickmotor (Copilotstickgehäuse)....38
Abbildung 3.7 : Inkrementalgeber Mechanik am Nickmotor (Pilotenstickgehäuse)....38
Abbildung 3.8 : Inkrementalgeber Mechanik am Rollmotor (Pilotenstickgehäuse).... 38

Abbildung 4.1 : Grundalgorithmus zum Abruf der Winkeldaten....40
Abbildung 4.2 : Geschwindigkeitssignal Generierung....41
Abbildung 4.3 : Sticksoftware Prozesse.... 43
Abbildung 4.4 : Sticksoftware Prozesse (mit INK und Koppel Ergänzung)....47

Tabelle 3.1 : Aufwandstabelle für Grundprinzipien 29
Tabelle 3.2 : Bewertungstabelle....33

1 Einführung

1.1 Der Forschungssimulator



Abbildung 1.1 : Der Forschungssimulator

Im Laufe von mittlerweile fast zehn Jahren ist am Institut für Flugmechanik und Regelungstechnik (FMRT) ein einmaliger Forschungssimulator eines Verkehrsflugzeuges entstanden, der mit einem 180° Sichtfeld nicht nur ein sehr realistisches Fluggefühl bietet, sondern mit seinen modular aufgebauten Bedien- und Anzeigeinstrumenten auch die Grundlage für weitere Forschungen in nahezu allen Bereichen der Flugzeugführung bietet. Dazu gehören unter anderem auch die Untersuchung der Mensch-Maschine Schnittstelle, welche heutzutage bei der immer größer werdenden Informationsflut im Cockpit, aber auch der zunehmenden Unterstützung des Piloten durch Computer, immer wichtiger wird.
Eine der wohl wichtigsten Mensch-Maschine Schnittstellen in einem Cockpit ist heutzutage immer noch der „Steuerknüppel“. Dieser ist im Gegensatz zu ältern Verkehrsflugzeugen allerdings nicht mehr eine kraftschlüssige Verbindung zwischen der Hand des Piloten und den Rudern des Flugzeuges, sondern nur noch ein Eingabegerät für den Flugsteuerungsrechner. Er bewirkt lediglich die Veränderung eines elektronischen Signals zur Weiterverarbeitung in der Flugsteuerung und wird deshalb heutzutage „Sidestick“ genannt.
Um so wichtiger ist es heute, sich um diesen Sidestick bezüglich der Mensch-Maschine-Schnittstelle zu kümmern, da durch dieses mechanische Abschneiden des Steuerknüppels zum Sidestick wichtige Informationen wie z.B. Steuerdrücke und damit verbunden das Gefühl für den Flugzustand der Maschine verloren gehen.
Um diese wichtigen Informationen dem Pilot wieder geben zu können, wurde vor ca. fünf Jahren der „aktive Sidestick“ entwickelt. Dabei handelt es sich um einen Stick, dessen Winkel / Drehmoment Kennlinie durch Torque Motoren (drehmomentstarke, elektronisch kommutierte Reluktanzmotoren) bestimmt wird. Diese Motoren übernehmen somit auch die Zentrierung des Sticks und ersetzten damit eine Rückstellfeder, wie sie bei passiven Sticks vorhanden ist.
Der Vorteil dieses aktiven Sidesticks sind die über Software programmierbaren Federkennlinien (Winkel / Drehmoment Kennlinie), welche auch flugzustandsabhängig variiert werden können. Dadurch bekommt der Pilot wieder die haptische Rückkopplung

für den Flugzugstand und darüber hinaus können noch weitere haptische Informationen wie z.B. Warnungen übermittelt werden.
Wie das System des aktiven Sidesticks in der Praxis realisiert wird und welche Komponenten dabei benötigt werden, soll im folgenden Kapitel erörtert werden.

1.2 Einbindung der Stickrechner im Forschungssimulator

Der Forschungssimulator besteht aus vielen einzelnen Systemen und Prozessen, die alle fast ausschließlich über den Datenpool kommunizieren. Der Datenpool ist eine Art zentrale Schnittstelle, in dem die Prozesse Daten zur Verfügung stellen oder abrufen und in dem auch die Winkel der Sidesticks kontinuierlich aktualisiert werden. Da die StickPCs voll mit der Stick-Regelung ausgelastet sind, übernimmt die Poll-Kommunikation ein Hochleistungsrechner Namens „Picard“, welcher über ein Y-Kabel die Messdaten der Winkel genau wie die StickPCs erhält und in einer VME Bus Karte A/D wandelt. Um die Netzwerklast möglichst gering zu halten, sind die StickPCs über ein eigenes lokales Netzwerk miteinander verbunden, mit dem nur noch ein dritter Rechner namens „Data“ verbunden ist. Dieser übernimmt die Kommunikation vom Pool an die StickPCs, wenn z.B. die Kennlinie flugzustandsabhängig verändert werden soll. Hierzu sind spezielle Tools von Thomas Gehler programmiert worden, die über die

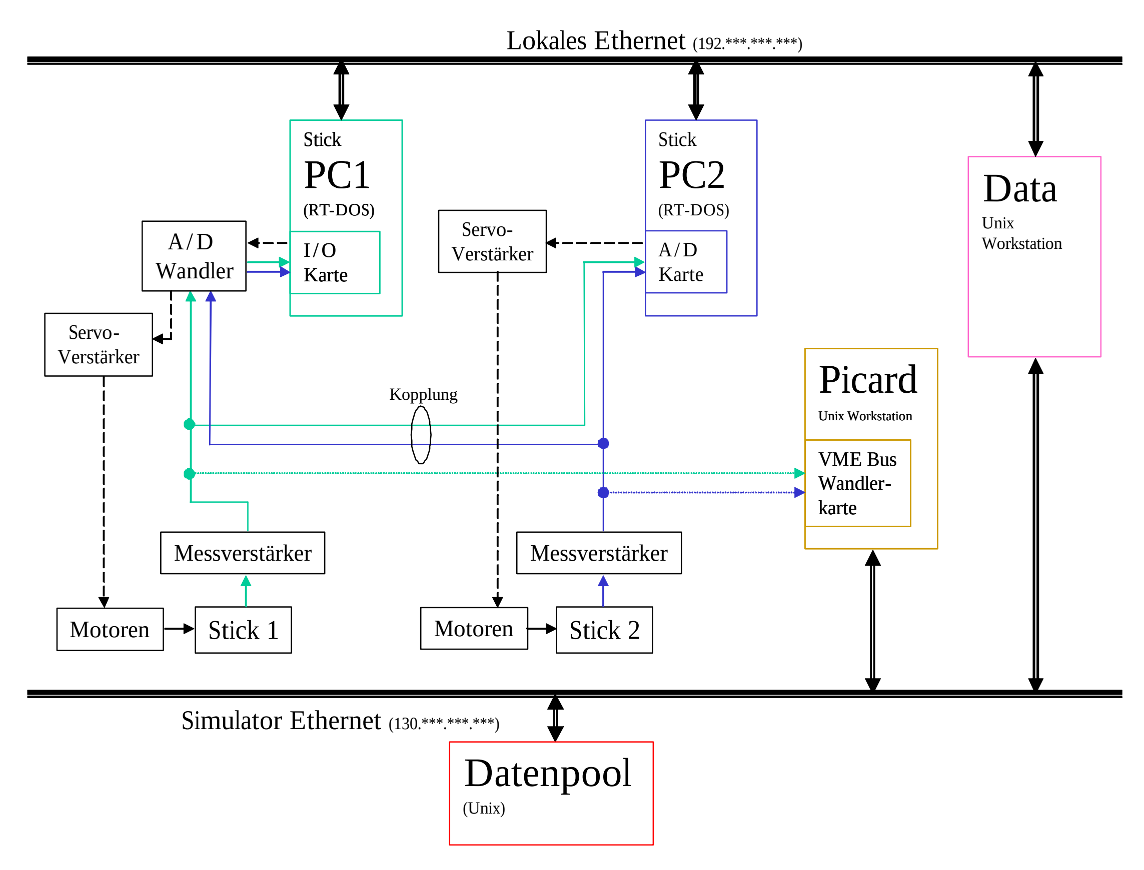


Abbildung 1.2 : Einbindung des Sticksystems

speziell entwickelte Kennliniensprache der StickPCs und das Programm „Pekas“ den Datenaustausch zwischen dem Pool und den StickPCs ermöglichen. Genauere Dokumentation dazu siehe Diplomarbeit von Thomas Gehler [1] Seite 25-30.
Die Abbildung 1.2 soll die StickPCs und die erläuterte Kommunikation mit dem Datenpool verdeutlichen. Die weiteren Stickkomponenten werden im folgenden Kapitel erörtert.

1.3 Das Sticksystem

1.3.1 Mechanik und Motoren

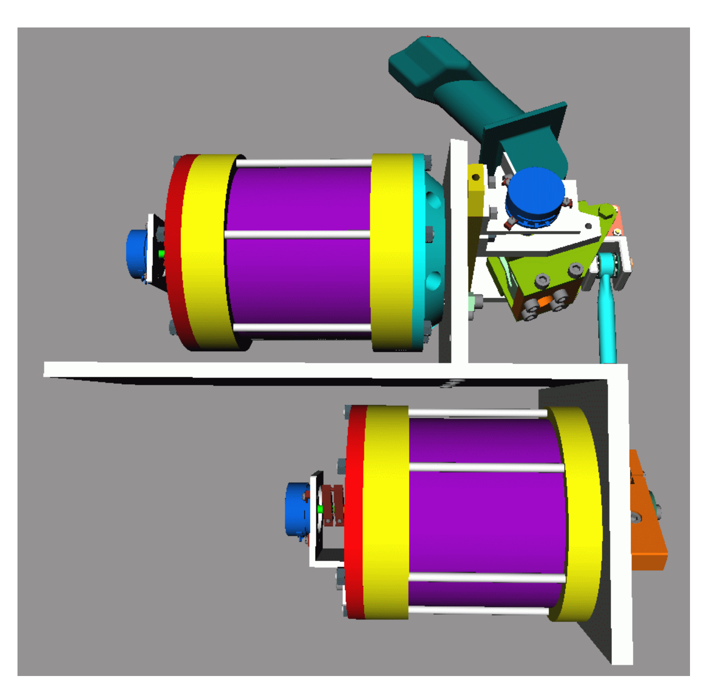


Abbildung 1.3 : Die Stickmechanik

Wie die Abbildung 1.3 zeigt, sind die größten Bauteile der Sticks die Torque Motoren mit folgenden technischen Eckdaten:

Typ	elektronisch kommutierter Hohlwellenmotor RBE 03005-A00
Hersteller	Maccon
Nennspannung	150 V
Nennstrom	5,5 A
Max. Drehmonent	32,4 Nm
Nenndrehzahl	610 UpM
Baugröße	127 mm Außendurchmesser, 105 mm Länge
Gewicht	4,9 kg
Übersetzung der Mechanik zum Stick	Rollachse 1:1 Nickachse 1:1,2

Die Mechanik der Sticks besteht wie in Abbildung 1.3 zu sehen ist lediglich aus Hebelarmen mit einem Gestänge und der Stickaufhängung, so dass das Drehmoment der Motoren auf den Stick mit nur geringem oder gar keinem Übersetzungsverhältnis übertragen wird.

Der wichtigste bauliche Unterschied zwischen dem Piloten- und Copilotenstick besteht hier vorallem in der Lagerung der Gestänge und der Stickaufhängung. Während der Pilotenstick (welcher als erster aufgebaut wurde) mit relativ normalen Kugellagern ausgerüstet ist, ist der Copilotenstick mit Wälzlagern konstruiert worden, welche sich durch ihre sehr geringe Reibung auszeichnen. Dieser Reibungsunterschied ist beim Bedienen der Sidesticks auch deutlich zu spüren und wirkt sich auch erheblich auf die Stabilität des jeweiligen Sticks in der Regelung aus.

1.3.2 Sensoren

Folgende Messdaten werden beim Stick kontinuierlich gemessen:

Signal	Messsensor
Nick –Winkel	Potentiometer
Roll –Winkel	Potentiometer
Nick – Drehmoment	Dehnmessstreifen
Roll – Drehmoment	Dehnmessstreifen
Strom Motorwicklung 1	Stromsensor in Servoverstärkern
Strom Motorwicklung 2	Stromsensor in Servoverstärkern

Die Winkeldaten werden in der Sticksoftware für den Regelalgorithmus der Position des Sidesticks benötigt. Über ein Y-Kabel direkt nach den Messverstärkern gelangen die Signale auch noch zu dem Hochleistungsrechner „Picard“ (Abbildung 1.2), der die Daten über den Simulator-Pool den Simulationsprozessen zur Verfügung stellt.

Die Drehmomentsignale werden derzeit für keine Regelung benutzt, sind aber für die Kopplung der Sticks notwendig (siehe 2.4.2).

Die Messwerte der Stromsensoren dienen vorallem zur internen Regelung der Servoverstärker, sind jedoch auch in der Software der Stick PCs abrufbar.

1.3.3 Elektronik

Messverstärker

In den Stickgehäusen wird die gesamte Messwertaufbereitung übernommen. Dabei werden Messsignale durch analoge Operations-Verstärkerschaltungen auf ein Spannungsniveau von ± 15 V verstärkt.

A/D und D/A Wandler

Die verstärkten Messsignale werden über Kabel aus dem Simulator-Cockpit zu den 16 Bit A/D Wandlern geführt. Beim StickPC1 werden die digitalisierten Messwerte anschließend über eine digitale Input/Output (I/O) Karte zur Messwertaufnahme im PC übertragen.
Beim StickPC2 wird eine neuere A/D-D/A Karte verwendet, welche direkt im PC eingebaut ist. Die Wandelfrequenz der Karten liegt bei max. 10 kHz, wobei sie jedoch aufgrund der beschränkten CPU Leistung der StickPCs durch die Software auf konstante 1 kHz gehalten wird.

Servoverstärker

Im Schaltschrank des StickPCs integriert, liefern sie max. 10 A bei ±50 V pro Verstärker um über Leitungen mit großem Querschnitt den Stickmotoren ihr Drehmoment zu geben. Die Berechnung der elektronischen Kommutierung der Motoren wird dabei von dem StickPC übernommen, der die jeweils nötigen Ströme der insgesamt drei Wicklungen über die D/A Wandler an die Servoverstärker übermittelt. Die Servoverstärker sorgen mit einem eigenen Regelkreislauf dafür, dass dieser vom PC vorgegebene Strom auch eingehalten wird. Sie wurden vom Hersteller der Motoren mitgeliefert und sind so optimal für die großen Ströme dimensioniert.

1.3.4 Software

Beim Sticksystem werden insgesamt bis zu 8 Kanäle mit einer konstanten Frequenz von 1 kHz eingelesen. Die Messwerte müssen während dieses Zeitraumes in einem Regelalgorithmus in Echtzeit berechnet werden um anschließend mit der gleichen Taktfrequenz die Motoren über die D/A Wandler anzusteuern. Um dies zu gewährleisten, muss ein echtzeitfähiges System diese Aufgaben bearbeiten.
Als vor ca. sieben Jahren damit begonnen wurde das System zu entwickeln standen nur wenige echtzeitfähige Betriebssysteme zur Verfügung, die auf herkömmlichen Intel 8086 Prozessoren basierenden PCs funktionierten. Daher wurde damals entschieden, Microsoft Dos Betriebssystem als Basis zu verwenden und es durch einen speziellen Real Time Kernel Aufsatz echtzeitfähig zu erweitern.

Real Time Programmierung

Mailboxen

Bei der Echtzeit Programmierung werden so genannte „Mailboxen“ verwendet, die eine Variablenübergabe zwischen einzelnen Tasks ermöglicht, ohne dass ein expliziter

Taskwechsel erfolgt. Damit können mehrer Task gleichzeitig den Prozessor nutzen und ihre Daten untereinander austauschen.

Prioritäten

Um zu gewährleisten, dass wichtige Tasks nicht durch unwichtigere Tasks blockiert werden, bekommt jede Task einen Prioritätsstatus, der ihr den Zugriff zum Prozessor entsprechend zusichert (hohe Priorität) oder auch stellenweise verweigert (niedrige Priorität). Der Regelalgorithmus hat beispielsweise die höchste Priorität, während die Keyboardeingaben nur mit niedriger Priorität arbeiten.

Echtzeit

Weiterhin kann durch den Real Time Kernel eine exaktes Zeitintervall vorgegeben werden, in der die Messwerte garantiert abgerufen werden sollen
Diese und noch weitere Features des Dos Real Time Betriebssystems sind in der RTK Broschüre des Herstellers [6] und der in der Diplomarbeit von Thomas Gehler [1] genau beschrieben.

Auf eine weitere Erörterung der Sticksoftware wird an dieser Stelle verzichtet, da die relevanten Teile in Kapitel 4.2 genau beschrieben werden. Insgesamt ist die Sticksoftware auch zu komplex um sie vollständig in einer Arbeit beschreiben zu können. Weitere Dokumentation zur Sticksoftware sind in [1] enthalten.

1.4 Die Aufgabe

Der wesentliche Punkt der Aufgabenstellung ist es, die Kopplung der Sidesticks zu ermöglichen. Dies bedeutet, dass sich der jeweils gegenüber liegende Stick mitbewegt, falls einer der beiden Piloten an seinem Stick eine Steuerbewegung vorgibt.
Diese Kopplung macht deshalb Sinn, weil der Pilot und der Copilot bisher Steuerbefehle eingeben können, und dabei unter Umständen nicht wissen, dass der jeweils andere ebenfalls mitsteuert. Das Steuerungssystem bildet einfach den Mittelwert aus beiden Steuerbefehlen, was in kritischen Situationen unter Umständen fatale Folgen haben kann. Zwar gibt es einen „Overwrite“ Knopf an jedem Stick, welcher bewirkt, dass der, der ihn zuerst drückt, die volle Steuergewalt unabhängig von den andern bekommt, doch verstreichen unter Umständen wichtige Sekunden, bis dieser Knopf gedrückt wird.
So soll in Anlehnung an die „echten Steuerknüppel“, bei denen die Kopplung noch über ein rein mechanisches Gestänge erfolgt, dies auch für den Sidestick realisiert werden. Wie diese Kopplung genau realisiert werden kann, wird im Abschnitt „Regelungssystem bei der Kopplung“ genau erörtert.

Doch bevor die Aufgabe der Kopplung angegangen werden kann, ist noch eine Verbesserung des Verhaltens der Stick nötig. Denn bei dem bisherigen Verhalten der Sticks ist es möglich, bei der Eingabe gewisser, gar nicht so ungewöhnlicher Kennlinien, die Stick zum Aufschwingen zu bringen. Die Ursache dafür wird ebenfalls im Kapitel „Das Regelungssystem“ genau erörtert. Ein Stichwort soll aber bereits vorweggenommen werden: Es muss eine zusätzliche Dämpfung eingebaut werden, welche aber durch die Software realisiert werden soll. Dies bedeutete aus regelungstechnischer Sicht, dass ein Geschwindigkeitssignal in der Software generiert werden muss.

Um dieses Geschwindigkeitssignal zu erhalten, werden einfach in einem definierten Zeitintervall zwei Positionen gemessen, wobei die Differenz dann die Geschwindigkeit ergibt. Die Realisierung mit den bisherigen Winkelgebern (Schleif-Potentiometern mit Analogverstärkern) ist aber aufgrund der Signalungenauigkeit nicht wünschenswert, so dass bereits im Vorfeld der Studienarbeit fest steht, dass zusätzliche Positionsgeber in den Stick eingebaut werden sollen.

Diese Positionsgeber waren bereits von den Entwicklern des Sticks ausgesucht und gekauft worden, so dass nur noch die mechanische Implementierung und das Programmieren des Computerinterface innerhalb der Studienarbeit anstehen. Bei den Sensoren handelt es sich dabei um Inkrementalgeber, welche mit 1024 Strichen auf 360 Grad und mit einer zusätzlichen Softwareinterpolation von noch einmal 1024 Stellungen eine realistischen Auflösung von 3,4 Milligrad haben. (Technische Daten und Berechnungen siehe Anhang III.1)

Das Computerinterface war eine eigens für diese Serie der Inkrementalgeber geliefert PC Karte mit ISA Bus, für die auch schon erste Programmierbeispiele vorliegen.

Doch nicht nur wegen der Dämpfungsaufgabe sollen die Inkrementalgeber eingebaut werden, sondern für die Zukunft wird auch schon daran gedacht, die derzeitigen Winkelsensoren durch die viel genaueren und deutlich einfacher zu implementierenden Inkrementalgebern vollständig zu ersetzen. Daher muss bei dem Einbau der Inkrementalgebern auch auf höchste Präzision und Spielfreiheit geachtet werden.

<u>Zusammenfassend ergeben sich damit für die Studienarbeit folgende Aufgaben:</u>

- Konstruktion und Realisierung einer Mechanik zum Einbau der Inkrementalgeber in jeden der beiden Sticks (zwei Inkrementalgeber pro Stick)
- Regelungstechnische Analyse der einzelnen Sticksysteme und der gekoppelten Sticks
- Einbindung eines Dämpfungssignals in die Sticksoftware durch Programmierung der Inkrementalgeberkarte
- Realisierung der Kopplung beider Sticks

2 Problemanalyse

2.1 Das Regelungstechnische Blockschaltbild

Um die auftretenden Instabilitäten auf theoretischer Basis erfassen zu können, ist es notwendig, das gesamte System mit allen seinen Bestandteilen regelungstechnisch zu erfassen. Hierzu werden einige Test an dem System vorgenommen und Daten aus den Dokumentationen entnommen, um schließlich folgendes Blockschaltbild aufstellen zu können.

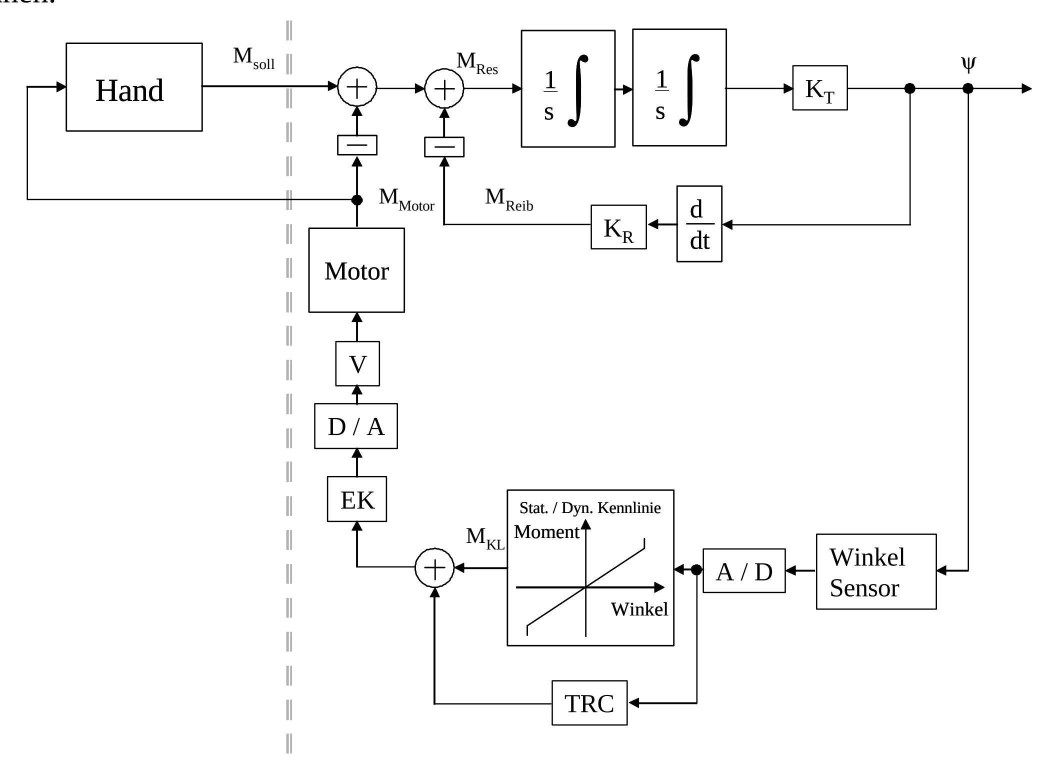


Abbildung 2.1 : Regelungstechnische Blockschaltbild (Grundsystem)

Legende zu Abbildung 2.1

Größe	Wert	Beschreibung des Blockes	Eingang	Ausgang
K_T [1)]	R 1 / 0,0125 kgm^2 N 1 / 0,0115 kgm^2	Kehrwert des Trägheitsmoments von Motor + Mechanik + Stick	$\iint M\, d^2t$	ψ $[°]_A$
K_R [3)]	ca. 0,6 Nms/° (stickabhängig)	Reibung der mechanischen Komponenten	ω $[°/s]_A$	M $[Nm]_A$
Winkel Sensor [3)]	0,5 V/°	Winkelsensoren (Potentiometer) mit analogen Verstärkern	ψ $[°]_A$	U_ψ $[V]_A$
K_L [2)]	Variabel(0-5)Nm/°	Kennliniensteigung	ψ $[°]_D$	M $[Nm]_D$
TRC [2)]	Dynamisch	Torque Ribble Kompensation	ψ $[°]_D$	M $[Nm]_D$
EK [2)]	ca. 0,2 V/Nm	Elektronische Kommutierung des Motors	M $[Nm]_D$	U_s $[V]_D$
V [1)]	0,67 A/V	Verstärkung der Servoverstärker	U_s $[V]_A$	I $[A]_A$
Motor [1)]	2,35 Nm/A	elektronisch kommutierter Torque Motor	I $[A]_A$	M $[Nm]_A$

Zeitkonstanten

T_V [2)]	1 ms	D / A Wandler zu Servoverstärkern
T_ψ [2)]	1 ms	A / D Wandler des Winkelsignals (der Potentiometer)
T_I [2)]	1 ms	A / D Wandler des Inkrementgebers
T_M [2)]	1 ms	A / D Wandler des Drehmomentsignals (DMS)
T_{array} [2)]	0-30 ms	Software generiertes Zeitintervall für Ableitung des Inkrementalgebersignals

Erklärungen:
[1)] aus Studienarbeit von Alexander Steiner [2]
[2)] durch Software definiert
[3)] eigene Messungen oder Berechnungen

Abkürzungen: R => Rollachse, N => Nickachse
Zeichenerklärung:
$[Nm]_D$ ist ein Drehmoment, welches digital erfasst wurde.
$[Nm]_A$ ist ein Drehmoment, welches analog bzw. mechanisch vorliegt.

2.1.1 Erörterungen zum Blockschaltbild

Die Hand

Als besonderen Grundbaustein soll an erster Stelle die Hand erwähnt werden, da sie auch in der Legende nicht genannt wird. Dazu ein kurze Erklärung: Um die Instabilität eines Systems zu betrachten ist es in der Regelungstechnik üblich, bestimmte extreme Sollwertvorgaben (z.B. eine Sprungfunktionseingabe) vor zu geben und dann zu messen, wie das System darauf reagiert. Die menschliche Hand macht aber gerade hier an dieser haptischen Schnittstelle nicht nur eine Sollwerteingabe, sondern reagiert auch wieder sofort auf die Antwort des Systems. Die menschliche Hand ist daher ein eigener Regelungskreislauf für sich, da sie zusammen mit dem „Rest des Körpers“ sowohl Aktor, als auch Sensor ist. Daraus ergeben sich wieder ganz andere Probleme, wie z.B. Piloten induzierte Schwingungen (PIO), welche auch in [1] beschrieben werden.
Für die benötigte Stabilitätsanalyse des Systems muss jedoch die Reaktion der Hand nicht mitbestimmt werden, da sie bei den gegebenen Schwingungsfrequenzen näherungsweise immer als eine Dämpfungsgröße beschrieben werden kann und daher sogar ein stabileres Verhalten des Gesamtsystems bewirkt. Deshalb wird bei den Untersuchungen die Reaktion der Hand vernachlässigt und das System mit Sprungfunktionen des Sollwertes auf Stabilität überprüft.
In der Praxis wäre dies gleichbedeutend mit dem Auslenken des Sticks auf den maximalen Winkel einer Achse und dem anschließenden Loslassen, so dass der Stick durch die Zentrierwirkung der Kennlinienvorgabe wieder in die Mitte beschleunigt. Und

dies ist in der Tat auch der Vorgang, um in der Praxis das Aufschwingen des Systems heraus zu fordern.
Die gestrichelte vertikale Abgrenzungslinie in Abbildung 2.1 soll das Vernachlässigen der Hand als Regelkreislauf auch im Blockschaltbild deutlich machen.

TRC

Die Torque Ribble Kompensation stellt für die Instabilität des Systems keine entscheidende Rolle dar und ist nur aufgrund der Vollständigkeit in den Blockschaltbildern skizziert. Die TRC ist ein dynamisch eingreifender Prozess, der je nach Winkelstellung des Stick zusätzlich geringe Drehmomentvorgaben macht, um die Drehmoment Schwankungen des Motors zu glätten. Siehe dazu auch [1].

Die Intergrationsglieder:

In der Physik gilt folgender Zusammenhang zwischen Drehmoment und Winkel, welcher bei der Stickregelung einen wichtigen Teil der Regelstrecke darstellt:

$$M = T \cdot a = T \cdot \frac{d^2 ?}{dt} \qquad (2.1)$$

Daraus resultiert für den Winkel:

$$? = \frac{1}{T}\iint M\,dt\,dt = \frac{1}{T}\int (M \cdot t)\,dt + v_0 = \frac{1}{T}\left(M \cdot t^2 + v_0 t + ?_0\right) \qquad (2.2)$$

somit ist die Winkelstellung das Resultat aus einem Drehmoment, mit den Anfangsgrößen „Anfangs-Winkel“ und „Anfangs-Geschwindigkeit“

Auf die Stickregelung bezogen bedeutet dies, dass das resultierende Moment aus den am Stick angreifenden Momenten die Winkelveränderung verursacht.
Der Hauptsummationspunkt lautet daher:

$$M = M_{res} = M_{soll} - M_{Motor} - M_{reib} \qquad (2.3)$$

2.2 Bestimmung der Übertragungsfunktion

Um das Regelungssystem auf mathematischer Grundlage beschreiben zu können, wird die Abbildung 2.1 auf Abbildung 2.2 vereinfacht:
Dazu werden die Proportionalglieder sinnvoll zusammengefasst.
Weiterhin werden im ersten Schritt die A/D Wandler vernachlässigt.

Es ergibt sich somit folgendes Blockschaltbild:

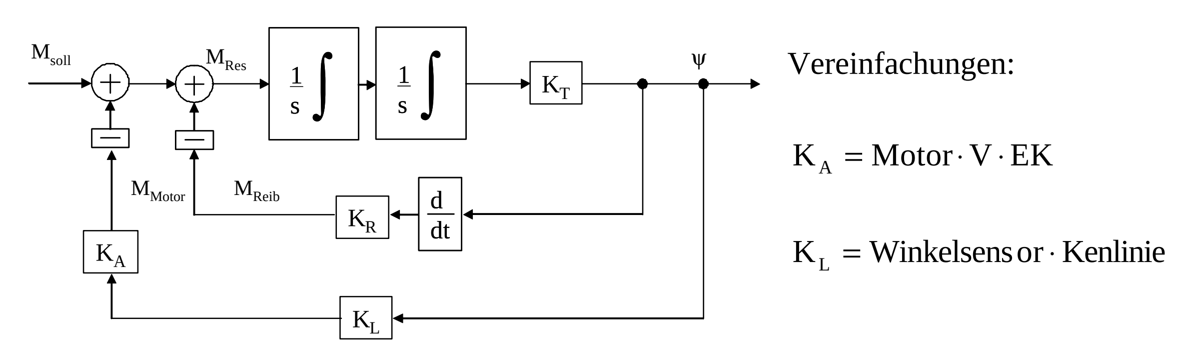


Abbildung 2.2 : Blockschaltbild vereinfacht (ohne A/D Wandler)

Die innere Kreisschaltung kann zusammengefasst werden, so dass sich ergibt:

$$F_i = \frac{1}{sK_R + s^2 \frac{1}{K_T}} \qquad (2.4)$$

Die äußere Kreisschaltung kann anschließend zusammengefasst werden, so dass sich die Führungsübertragungsfunktion des Regelungskreislaufes ergibt:

$$F_W = \frac{1}{K_A K_L + sK_R + s^2 \frac{1}{K_T}} = \frac{\frac{1}{K_A K_L}}{1 + s\frac{K_R}{K_A K_L} + s^2 \frac{1}{K_T K_A K_L}} \qquad (2.5)$$

Dies entspricht einem PT_2– Glied !!

Allgemeine Darstellung PT_2: $\frac{K}{1 + sT_0 + s^2 T_0 T_1}$ mit $d = \frac{1}{2}\sqrt{\frac{T_0}{T_1}}$; $\nu_0 = \frac{1}{\sqrt{T_0 T_1}}$

2.2.1 Stabilitätsanalyse

Da die Pole des PT_2 Gliedes bei positiven Koeffizienten immer einen negativen Realteil haben, ist das PT_2 Glied grundsätzlich stabil.
Die einzige Art der Instabilität liegt vor, wenn die Dämpfung vollständig verschwindet, so dass sich eine oszillatorische Instabilität ausbilden kann.
Dies wäre im Sticksystem der Fall, wenn die mechanische Reibung $K_R = 0$ bzw. das Trägheitsmoment $K_T = 0$ ist, was jedoch in der Praxis nicht vorkommt.
Siehe dazu Formel (2.6)

$$d = \frac{1}{2}\sqrt{\frac{K_T K_R^2}{K_A K_L}} \qquad (2.6)$$

Um nun zu verstehen, warum sich das System dennoch in der Praxis aufschwingt, muss auch der Einfluss der A/D Wandler betrachtet werden.

Die A/D Wandler

Um das System weiterhin als ein kontinuierlichen System beschreiben zu können, ist es auch zulässig die A/D bzw. D/A Wandler als Verzögerungsglieder zu betrachten, da sie sozusagen den Fluss der Information um genau ihre Abtastzeit verzögern. Vorraussetzung für die Vereinfachung ist jedoch, dass sich innerhalb einer Abtastperiode die Amplitude nur sehr geringfügig verändert, damit das Diskretisieren der Werte nicht ins Gewicht fällt. Da im Sticksystem die Abtastfrequenz bei 1kHz liegt, die Systemfrequenzen im Betrieb jedoch unter 100 Hz liegen, ist diese Vorraussetzung erfüllt.

Weiterhin kann aus der Regelungstechnik die Vereinfachung getroffen werden, dass ein Verzögerungsglied für Frequenzen unterhalb der Totzeit als PT_1– Glied angenähert werden kann. Somit gilt:

$$e^{-sT} \approx \frac{1}{1+sT} \qquad (2.7)$$

Mit diesen Vereinfachungen ergibt sich nun das neue vereinfachte Blockschaltbild:

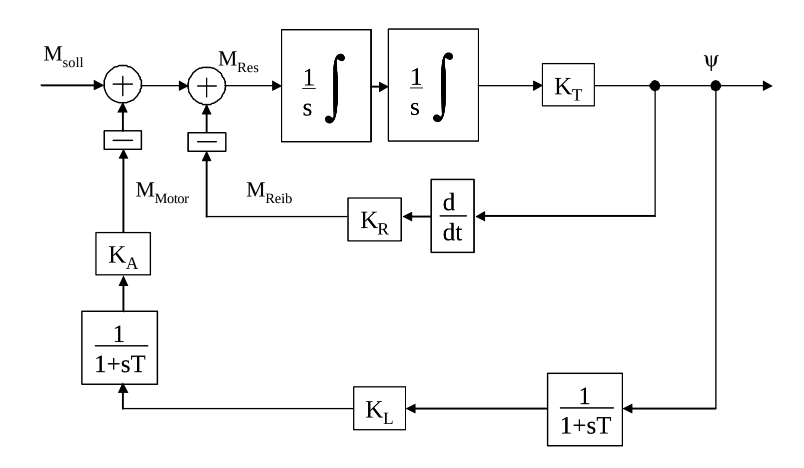


Abbildung 2.3 : Blockschaltbild vereinfacht (mit A/D Wandler)

Berechnet man nun die Führungsübertragungsfunktion für das mit den A/D-D/A Wandlern ergänzte System, so ergibt sich ein mathematisch nur schlecht anschauliches

Übertragungsverhalten. Daher wird an dieser Stelle das Simulationstool „Simulink“ (Teilprogramm von Matlab) benutzt, um weitere Aufschlüsse über das Übertragungsverhalten zu bekommen.
(Benutzte Blockschaltbilder zu Simulink siehe Anhang I.3)

Ergebnis der Simulation mit Simulink:

Die Instabilität des Systems wird durch die Verzögerungsglieder verursacht, da sie zusätzlich zum Phasengang des PT_2-Gliedes, die Phasenverschiebung bei der Rückkopplung erhöhen. Dadurch kann es zu einem Phasengang von über -180 Grad bei einem Betrag von größer 1 führen, was nach dem Nyquist-Kriterium (siehe [3] S 83) charakteristisch für instabiles Verhalten ist.

Schlussfolgerung der Stabilitätsproblematik

Da es in dem Sticksystem derzeit keine Möglichkeit gibt, die Verzögerungsglieder bzw. die A/D-D/A Wandler zu verbessern, muss die verursachte Phasenverschiebung durch eine zusätzliche Dämpfung wieder kompensiert werden.
Somit muss nun die Wirkung einer Dämpfung in dem System untersucht werden, wie es auch bereits in der Aufgabenstellung der Studienarbeit gefordert wird.

2.3 Analyse der zusätzlichen Dämpfung

Die Implementierung eines zusätzlichen Sensors (Inkrementalgebers) mit entsprechender I/O Karte im StickPC und einer Software generierten Ableitung, wird in der folgenden Abbildung dargestellt:

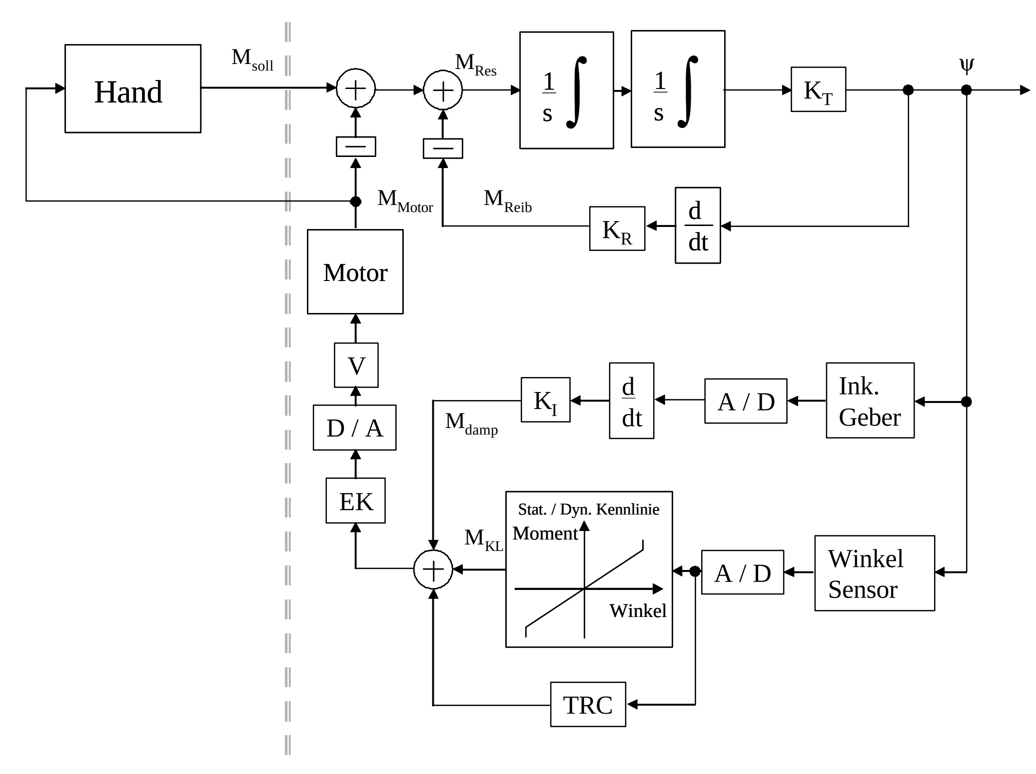


Abbildung 2.4 : Regelungstechnische Blockschaltbild (mit Inkrementalgeber)

Ergänzende Legende (Erweiterung zu Legende Abbildung 2.1.):

Größe	Wert	Beschreibung des Blockes	Eingang	Ausgang
Ink. Geber	2913 Inkremente/° 50,84 Inkremente/rad	Inkremental Geber	ψ [rad]$_A$	U_ψ [V]$_A$
K_I	R 0,00059 * D Nms/° N 0,00049 * D Nms/°	Skalierwerte (konstant) * Dämpfungswerte (variabel) des Inkrementalgebers (zu D siehe Anhang I.1)	ψ [°]$_D$	M [Nm]$_D$

2.3.1 Bestimmung der neuen Übertragungsfunktion

Um die Auswirkungen der Ergänzung durch den Inkrementalsensor auch in der Übertragungsfunktion analysieren zu können, wird wie bereits im Abschnitt 2.2 vorgegangen. Dazu wird zuerst wieder Abbildung 2.4 vereinfacht:

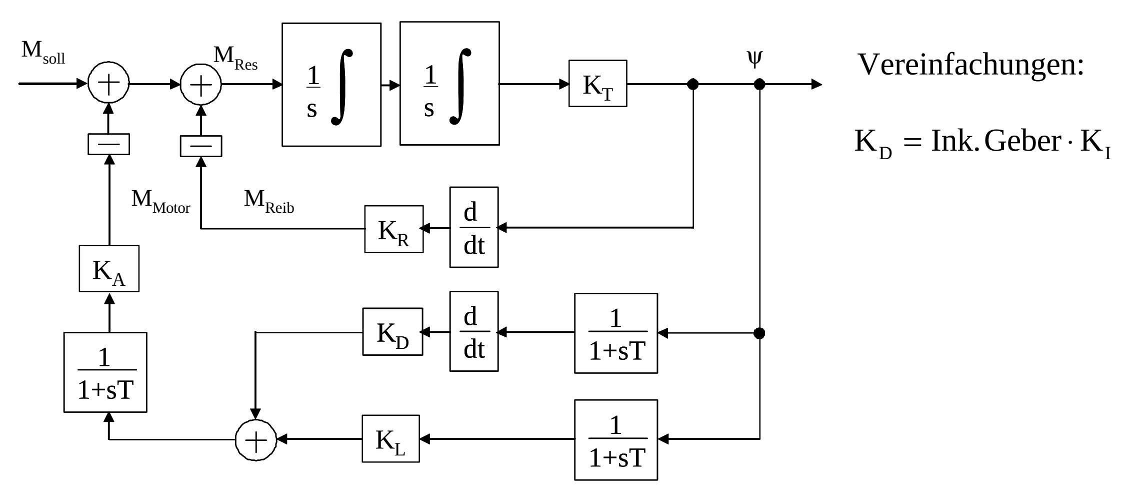


Abbildung 2.5 : Blockschaltbild vereinfacht (mit Inkrementalgebern)

Durch Zusammenfassung der Parallelschaltung in der Rückführung ergibt sich eine neue Rückführungsfunktion $F_{rück}$ (ohne Verzögerungsglieder !!):

$$K_A \cdot (K_L + sK_D) \tag{2.8}$$

Durch Zusammenfassung der oberen Kreisschaltung und der äußeren Kreisschaltung ergibt sich schließlich die neue Führungsübertragungsfunktion zu:

$$F_{W_n} = \frac{\frac{1}{K_A K_L}}{1 + s\frac{(K_R + K_A K_D)}{K_A K_L} + s^2 \frac{1}{K_T K_A K_L}} \tag{2.9}$$

Daraus resultiert eine veränderte Dämpfung des PT_2-Gliedes von:

$$d = \frac{1}{2}\sqrt{\frac{K_T (K_R + K_A K_D)^2}{K_A K_L}} \tag{2.10}$$

Im Vergleich zu Formel (2.6) wirkt nun auch das Inkrementalgebersignals zusätzlich zur mechanischen Reibung als Dämpfungsgröße.

Weitere Simulationen mit Simulink, in denen auch die Verzögerungsglieder berücksichtigt sind, zeigen, dass die zusätzliche Dämpfung die Phasenverschiebung der Verzögerungsglieder kompensieren kann.
Somit kann die Instabilität durch die zusätzliche Dämpfung der Inkrementalgeber behoben werden !

2.4 Problemanalyse der Kopplung

Ähnlich wie bei den Untersuchungen der einzelnen Sticks wird bei den Untersuchungen zur Kopplung beider Sticksysteme vorgegangen. Es wird in ersten Tests die bisher realisierte Kopplung untersucht, welche erhebliche Probleme der Instabilität und der ungewollten Kennlinien Veränderung aufweist. Anschließend wird ein Modell der Kopplung erstellt, welches die Fehler der bisherigen Kopplung und die Möglichkeit zur Verbesserung aufzeigt.

Modellbildung

Die folgenden Abbildungen sollen die Problematik der Kopplung von zwei unabhängigen Systemen darstellen. Dabei wird der Vergleich zwischen einem mechanischen Modell zu dem elektromechanischen Stickmodell gezogen.

Das Problem der elektromechanischen Kopplung:

Bei einem mechanischen System wird die Kopplung durch einen zusätzlichen Angriffspunkt einer Kraft am Stick realisiert. (siehe Abbildung 2.6)

Dies ist bei der elektronischen Kopplung nicht möglich, da nur der Angriffspunkt des Motors existiert und damit die Ansteuerung des Motor eine zweite Krafteinwirkung simulieren muss. Dies kann auch so interpretiert werden, dass der Motor ein Offset vorgegeben bekommt, welches von dem Stick gegenüber jeweils beeinflusst wird. (Dieses Offset kann zu Anschauungszwecken als mechanisches Offset dargestellt werden. (Siehe Abbildung 2.7))

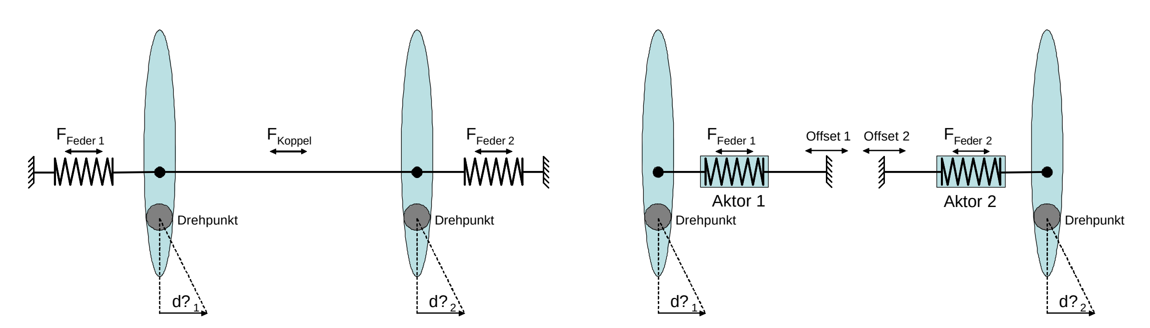


Abbildung 2.6 : mechanische Kopplung

Abbildung 2.7 : elektromechanische Kopplung

Damit stellt sich die Frage, aufgrund welcher Daten die Ansteuerung eine Kopplung simuliert wird. Dazu werden zwei Möglichkeiten erörtert:

2.4.1 Kopplungsprinzipien

<u>Winkelkopplung (bisher)</u>

Hier wird das Winkelsignal des gegenüber liegenden Sticks direkt an das Offset gebunden, was in der Modellbildung auch mechanisch dargestellt werden kann (siehe Abbildung 2.8).

Bei einer einseitigen Kopplung funktioniert dieses Prinzip sehr gut und die Federkennlinie beider Aktoren bleibt unverändert.

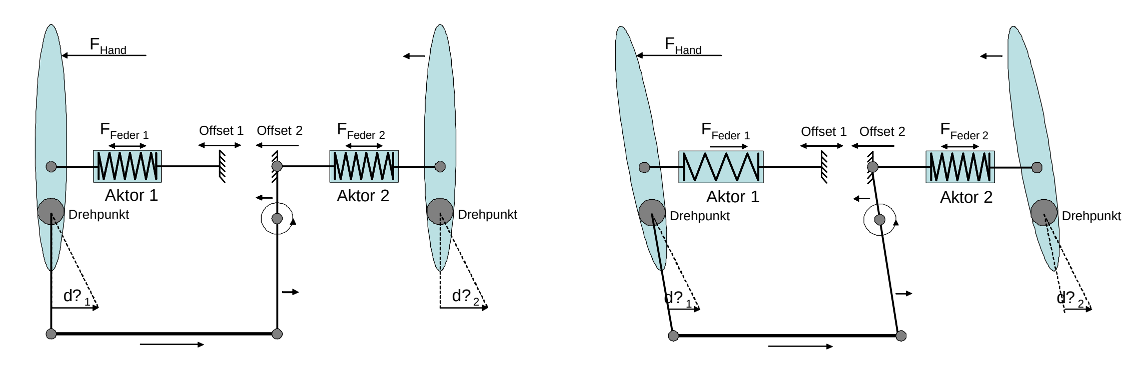


Abbildung 2.8 : einfache Winkelkopplung

Wird die Kopplung wechselseitig vorgenommen, so kommt es zu einer gegenseitigen Beeinflussung der Winkel und die Federkennlinie wird vollkommen unwirksam. Eine Winkeländerung des ersten Sticks hat dann eine Winkeländerung des zweiten Stick zur Folge, welche wieder zu einer Winkeländerung des ersten Sticks führt. (siehe Abbildung 2.9)

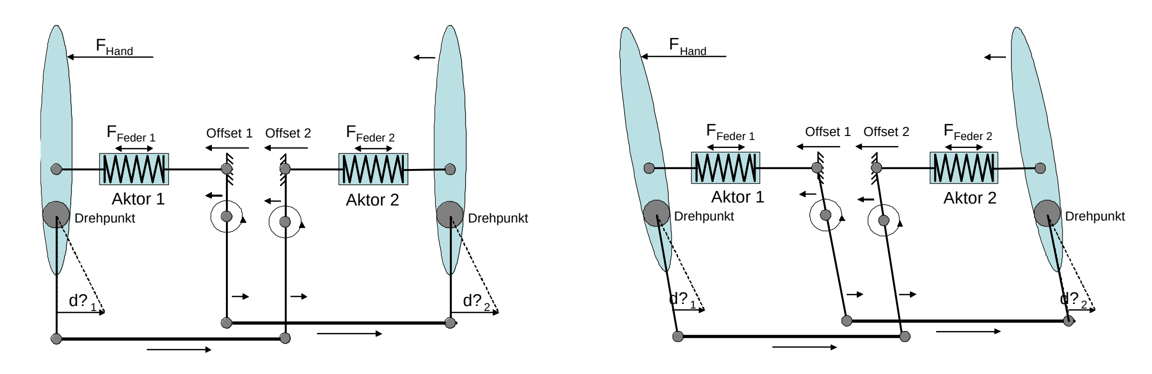


Abbildung 2.9 : doppelte Winkelkopplung

Dieses funktioniert jedoch nur rein theoretisch, denn in der Realität wird durch unvermeidbare Störgrößen bei einem Übersetzungsverhältnis von größer als eins diese doppelte Winkelkopplung vollkommen instabil. Nur für Übersetzungsverhältnisse von kleiner als eins ist eine Kopplung bedingt möglich, wobei jedoch die Federkennlinien massiv beeinflusst wird.

Damit sollte dargestellt werden, dass die doppelte Winkelkopplung, wie sie bisher realisiert ist, kein zufrieden stellendes Ergebnis liefern kann.

Um eine andere Kopplungsmöglichkeit zu finden, kann die Winkelkopplung auch als ein unterbestimmtes Gleichungssystem beschrieben werden. Um eine Kopplung zu realisieren, bei der sowohl die Federkennlinie als auch die Kopplung unabhängig voneinander funktionieren, muss somit eine weitere vom Winkel unabhängige Variable gemessen und in das System eingebunden werden.

Diese Variable kann z.B. das Drehmoment sein, welches bereits an den Sticks durch Dehn-Mess-Streifen (DMS) gemessen wird.

Drehmomentkopplung

Bei diesem Prinzip wird das Drehmoment, welches durch die Hand und die dadurch resultierenden Reaktion des Aktors hervorgerufen wird, zur Offset Verschiebung des gegenüber liegenden Sticks benutzt.
Wirkt nun eine Handkraft auf einen der Sticks, so wird durch das aufgebrachte Drehmoment der Winkel des anderen Sticks verstellt. Die Kennlinie von beiden Sticks ist dabei vollkommen unverändert.

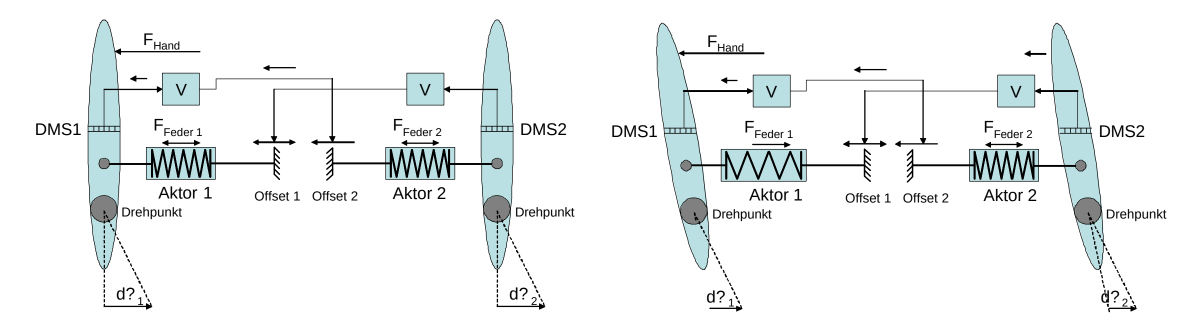


Abbildung 2.10 : doppelte Drehmomentkopplung

Wirkt nun auch noch eine Handkraft auf den zweiten Stick, so muss die Handkraft des ersten Sticks deutlich vergrößert werden, damit die Winkelstellung gleich bleibend stabil gehalten werden kann. Die Kennlinien beider Sticks bleiben bei dieser Art der Kopplung jedoch vollständig erhalten ! In der Praxis merkt somit der Pilot, dass jemand „gegen ihn steuert“.

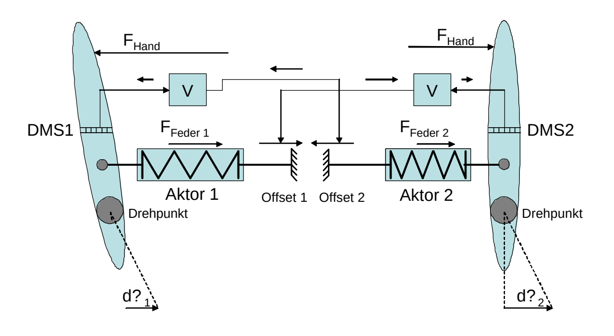


Abbildung 2.11 : zwei wirkende Handkräfte

Vergleich von mechanischer zu elektromechanischer Kopplung

In der Modellbildung wird klar, dass eine gegenseitige Winkelbeeinflussung der Sticks möglich ist, so dass der Pilot Steuereingriffe des Copiloten und umgekehrt spüren kann. Von einer Kopplung kann allerdings nur bedingt gesprochen werden, da es mit entsprechenden Kraftaufwand möglich ist, die Sticks auch mit unterschiedlichen Winkeln zu halten (siehe Abbildung 2.11). Dies ist jedoch bei einer echten mechanischen Kopplung nicht möglich, da dort die Winkel durch die mechanisch starre Verbindung immer die gleiche Position einnehmen (siehe Abbildung 2.6)
Es ergibt sich somit ein völlig neues Steuergefühl, welches erst auf seinen Gebrauchswert getestet werden muss. Aufgrund der Problematik der elektromechanischen Kopplung ist es jedoch nicht möglich, eine exakte mechanische Kopplung nach zu bilden.

2.4.2 Regelkreislauf der Kopplung

Um auch die Stabilität der Kopplung zu untersuchen, wird das bisherige Regelungstechnische Blockschaltbild wie folgt ergänzt:

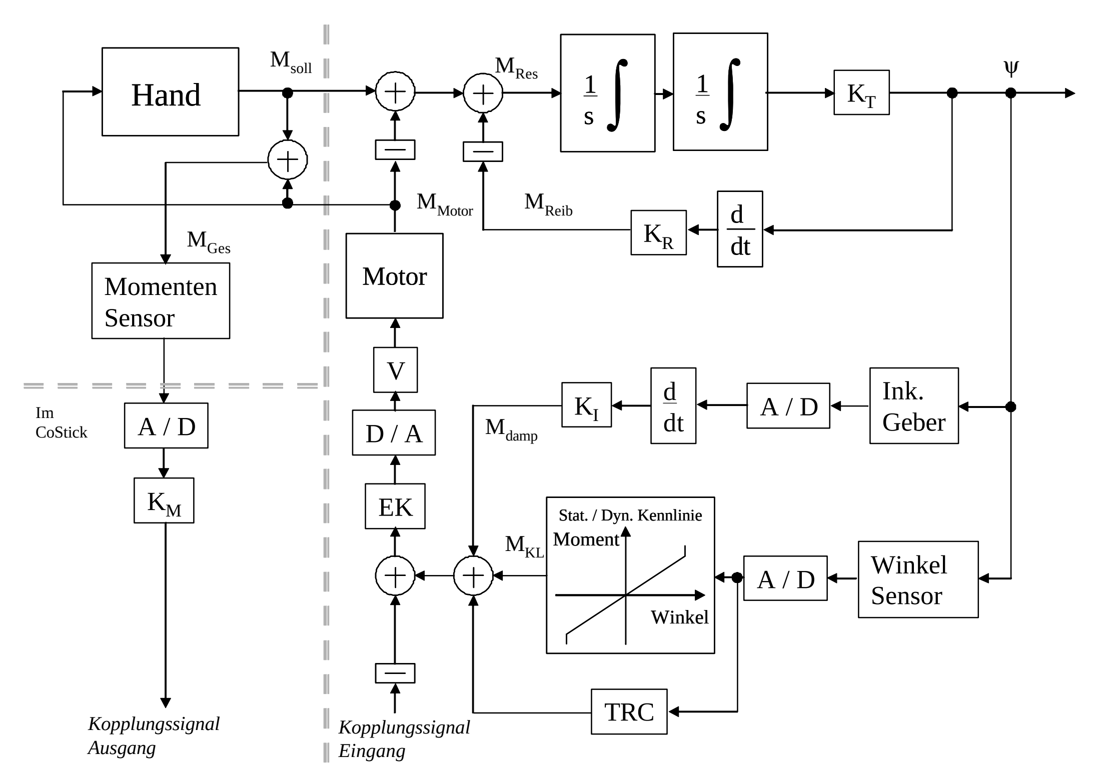


Abbildung 2.12 : Regelungstechnische Blockschaltbild der Kopplung

Ergänzende Legende (Erweiterung zu Legende Abbildung 2.1.):

Größe	Wert	Beschreibung des Blockes	Eingang	Ausgang
Momenten Sensor [3)]	1 V/Nm	Dehnmessstreifen (DMS) mit analogen Verstärkern	M [Nm]$_A$	U_M [V]$_A$
K_M	Variabel (0,1-1) Nm/V	Drehmomentskalierung	U_M [V]$_D$	M [Nm]$_D$

Wichtig!: Der Drehmomentsensor erfasst aufgrund seiner mechanischen Anordnung immer das Handmoment + das Motordrehmoment. Ist jedoch kein Handmoment vorhanden (Stick wird losgelassen) so wird auch kein Motormoment gemessen. In Abbildung 2.12 ist dies nur dann gegeben, wenn für den Zustand „keine Hand am Stick" alle Komponenten links der gestrichelten vertikalen Linie abgetrennt werden.

Die Stabilitätsuntersuchungen des gekoppelten Systems wird mit Simulink vorgenommen. Es zeigt sich, dass die Drehmomentkopplung keinen Einfluss auf die Stabilität der einzelnen Systeme hat und somit theoretisch einwandfrei funktioniert.
Die Winkelkopplung wird ebenfalls simuliert, mit den Ergebnissen wie sie in Abschnitt 2.4.1 beschrieben sind.

Alle Simulink Modelle und Aufzeichnungen dazu befinden sich im Anhang I.3

3 Inkrementalgeber Hardware

Im Abschnitt 3.1 wird auf die Konzeptbildung der Inkrementalgebermechanik eingegangen. Dazu wird in einem methodischen Entwicklungsprozess eine geeignete Mechanik ausgewählt, deren Realisierung dann in Abschnitt 3.2 beschrieben wird.
In Abschnitt 3.3 wird auf den Einbau der Inkrementalgeber I/O Karte eingegangen, welche für die Messwerterfassung im Sticksoftware benötigt wird.

3.1 Konzept Bildung

Bei der Suche nach Konzepten zum Einbau der Inkrementalgeber in das Stickgehäuse kommen folgende Fragestellungen auf:

1) Wie und Wo ist eine Bewegungsabnahme der Drehgeschwindigkeit am Motor/Stickmechanik möglich ?
2) Mit welcher Mechanik kann diese Bewegung an den Inkrementalgeber übertragen werden ?

Aus diesen Fragen resultiert eine Unterscheidung in zwei Bereiche bei der Bewertung:

1) Abgriff der Bewegung (Prinzip)
2) Kraftübertragung (Lösungsvarianten)

Dabei beschreibt der erste Teil unterschiedliche Prinzipien und Möglichkeiten, um einen ersten Eindruck über das Problem zu erhalten. Mit Hilfe der anschließenden ersten Bewertung mit relativ einfachen Bewertungskriterien entsteht dann eine Beurteilung dieser Grundideen.

Der zweite Teil beschreibt auf Basis der Grundprinzipien weiterführende Lösungsvarianten, welche bis ins Detail ausgearbeitet werden können.
Die Bewertung erfolgt im zweiten Abschnitt mit viel spezifischeren Bewertungskriterien, welche auf die Beurteilung der Unterschiede der einzelnen Lösungsvarianten zielt.
Die gesamte Bewertung entsteht schließlich aus der Addition der ersten Grundbewertung und der zweiten detaillierten Bewertung.
(Ein Blick auf die Bewertungstabelle (Tabelle 3.2) verdeutlicht diese Beschreibung)

3.1.1 Prinzipien

Die Funktion der Mechanik soll sein, eine rotatorische Bewegung abzugreifen und diese Bewegung an einem anderen Punkt rotatorisch am Inkrementalgeber wieder angreifen zu

lassen. Die Möglichkeiten dies zu erreichen, werden nun generell mit den folgenden Prinzipien dargestellt.

Prinzip 1: Translatorisch (Hebelarm der Welle)

Gedanke: Ein Hebelarm wird an einer Welle angebracht und übersetzt somit die rotatorische Bewegung für kleine Winkel (max. 90°) in eine quasi translatorische. Dieses Prinzip ist sowohl für den Aktor als auch für den Sensor zu nutzten.
(Anmerkung: die Drehwinkel der Sticks betragen max. 90°, wobei bei diesem Winkel die Strecke senkrecht zur gewünschten Bewegungsrichtung bereits relativ groß wird und somit deutliche Linearitätsfehler der Translationsbewegung auftreten. Es ist aber sehr gut möglich diese Fehler mit geometrischen Abhängigkeiten heraus zu rechnen, und so den Fehler rechnerisch zu korrigieren.)

Prinzip 2: Rotatorisch (an Potentiometerkupplung)

Hier geht es vor allem um die Frage an welchen Ort die Bewegungsaufnahme erfolgt. Hier wird speziell der Aufnahmepunkt an der Welle des Motors vorgegeben, an dem bisher die Potentiometer Kupplung sitzt. Eine Abnahme der Bewegung an diesem Punkt ist aus Platzgründen nur mit rotatorischen Prinzipien möglich.

Prinzip 3: Rotatorisch (an Stickmechanik Aufnahme)

Ebenso wie beim Prinzip 2 geht es hier um den Ort einer rotatorischen Bewegungsaufnahme: Hier wird ein Punkt gewählt, der an der Mechanik zu finden ist, die von der anderen Seite der Welle des Motors bewegt wird. Je nach dem ob der Nick- oder der Rollmotor betrachtet wird, ist es möglich, solch einen Punkt zu finden.

Bewertungskriterien

(Maximale Bewertung=5; minimale Bewertung=1; Ausschluß=0)

Aufwand:
Dieses Kriterium ist sehr allgemein verfasst, wobei man die Prinzipien auch nur sehr allgemein bewerten darf um die tatsächliche Auswahl erst unter Punkt zwei der Bewertung durchzuführen. Das Kriterium beschreibt, welcher Aufwand nötig ist, um das Prinzip zu realisieren. Dabei wird vor allem betrachtet, was alles an der bisherigen Mechanik verändert werden muss, um überhaupt das Prinzip realisieren zu können. Dazu wurde die Tabelle 3.1 aufgestellt, um einen ersten Vergleich zu schaffen und die anschließende Bewertung einsichtiger zu machen. Dabei wurde zusätzlich auch zwischen Roll- und Nickmotor/Mechanik unterschieden.
Die Bewertung 5 entspricht dabei einem niedrigen Aufwand, und die Bewertung 1 einem sehr hohen Aufwand.

Mechaniksymmetrie:
Bei diesem Punkt wird berücksichtigt, ob der Aufwand der für das Prinzip der Kraftanbindung an die Mechanik nötig ist, an beiden oder nur an einem der beiden Motoren möglich ist. Die Bewertung fällt um so schlechter aus, je mehr einzelner Aufwand für die beiden Motoren notwendig ist.

Platz am Abgriff:
Dieses Kriterium gibt einen Eindruck darüber wie viel Platz an dem Kraftaufnahmepunkt vorhanden ist. Die Beurteilung eines Prinzips erfolgt dabei in Relation zu den anderen Beiden.

	Nickmotor	Rollmotor
Prinzip 1 **Translatorisch** (Hebelarm an Welle)	Bisherige Mechanik müsste durch Mechanik mit Hebelarm ersetzt werden	Hebelarm bereit vorhanden
Prinzip 2 **Rotatorisch** an Potentiometerkupplung	Potentiometer-kupplung muss ausgetauscht werden	Potentiometerkupplung muss ausgetauscht werden
Prinzip 3 **Rotatorisch** an Stickmechanikaufnahme	Umbau des Gehäuses nötig, da zu wenig Platz	Nicht möglich

Tabelle 3.1 : Aufwandstabelle für Grundprinzipien

<u>Gewichtung der einzelnen Punkte:</u>

(Maximale Gewichtung=5; minimale Gewichtung=1)

Teil 1: Prinzipen
1) Aufwand => Gewichtung 5
Dieses Kriterium ist mit Abstand das wichtigste, da es vorrangig ist den Aufwand und damit auch den Zeitaufwand zu minimieren.

2) Mechaniksymmetrie=> Gewichtung 5
Ein ebenfalls wichtiger Faktor, da dadurch ebenfalls Zeit, Aufwand und damit auch Kosten minimieren werden können.

3) Platz am Abgriff => Gewichtung 4
Wichtig, damit die Konstruktion nicht zu aufwendig wird, weil nur sehr geringer Bauraum zur Verfügung steht.

3.1.2 Lösungsvarianten

Translatorische Prinzipen

Starre Verbindung

Sowohl die Bewegung des Motor, als die des Inkrementalgeber werden durch einen Hebelarm translatorisch übersetzt. Der Kraftschluss der Hebelarme erfolgt mit einer starren Verbindung.
Vorteile:
wenig bewegliche Teile
simple Konstruktion
gute Spielfreiheit
Nachteile:
unflexibel für die Positionsfindung des Inkrementalgebers im Stickgehäuse

Bowdenzug

Wie auch bei der starren Verbindung erfolgt beim Motor, als auch beim Inkrementalgeber, eine Bewegungsumsetzung ins Translatorische. Diese translatorische Bewegung kann dann vom einem Hebelarm zum anderen Arm mit einem Bowdenzug übertragen werden.
Vorteile:
flexiblere Positionsfindung des Inkrementalgebers im Stickgehäuse
Nachteile:
seitlicher Schlupf des Bowdenzuges in dem Führungsrohr führt zu Längenabweichungen.

Zahnstange an Hebelarm

Der Abgriff der Bewegung am Motor erfolgt wieder durch einen Hebelarm.
Die translatorische Bewegung wird an eine Zahnstange übertragen, welche schließlich ein Ritzel bewegt, dass an dem Inkrementalgeber befestigt ist.
Vorteile:
Platz des Hebelarmes an Inkrementalgeber wird deutlich verringert
Linearitätsfehler des Hebelarmes an Inkrementalgeber entfällt.

Nachteile:
aufwändiger zu konstruieren
Spiel in Zahnrädern
spezielle Teile nötig

Rotatorische Prinzipien

Zahnriemen

Das Prinzip eines Zahnriemenantriebes soll zwischen Motorwelle und Inkrementalgeberwelle angewandt werden.
Vorteile:
sehr wenig Schlupf => hohe Genauigkeit
fertig zu kaufende Teile
Nachteile:
Achsabstandsentfernung durch Zahnriemenlänge vorgegeben
Kurvenradius eingeschränkt durch Flexibilität des Zahnriemens

Seilzug

Das Prinzip des Zahnriemenantriebes wird hier weitergeführt, mit der Ergänzung, dass keine Zahnriemen und Zahnräder verwendet werden, sondern ein Seil und glatte Räder.
Vorteile:
Achsabstandsentfernung ist variabel
sehr kleiner Kurvenradius möglich
insgesamt kleinere Radabmessungen möglich
Nachteile:
Dehnschlupf des Seiles
extra anzufertigende Teile

Zahnstange mit Ritzeln

Hier wird die Idee der Zahnstange aus den translatorischen Prinzipien weitergeführt, wobei hier nun die Abnahme der Bewegung an beiden Achsen durch ein Ritzel erfolgen soll und somit die Zahnstange nur als Kraftschluss zwischen zwei Ritzeln funktioniert.
Vorteile:
variable Position des Inkrementalgebers
Nachteile:
Spiel der Ritzel

Bewertungskriterien

Platzbedarf der Mechanik
Beschreibt die Mindestgröße der Mechanik, die z.B. aufgrund von max. Biegeradien oder Steifigkeiten eingehalten werden muss. Ist eine Mechanik relativ klein realisierbar, erhält sie die maximale Bewertung. Die Entscheidungsgrundlagen für diesen Bewertungspunkt sind technische Erfahrungswerte.

Variable Position des Inkrementalgebers beim Einbau
Variable = +- 5mm um Messfehler und Einbautolleranzen zu kompensieren (bewertet wird die Variabilität => 5=hoch)

Spielfreiheit
Grundsätzlich: Um so mehr Kraftschlüsse vorhanden sind, um so größer ist auch das Spiel und um so schlechter die Bewertung
Zusätzlich wird jedoch auch die Art des Kraftschusses bewertet um spielarme von spielhohen Kraftschlüssen mit in der Bewertung zu unterscheiden.

Fertigungsaufwand
Abschätzung des Werkstatt- und eigenen Entwicklungs- und Einbauzeitraumes, der für die Realisierung der gesamten Mechanik notwendig ist.
Dabei besteht eine hohe Korrelation zur Komplexität und Anzahl der zu fertigenden Teile.

Gewichtung der einzelnen Punkte:

(Maximale Gewichtung=5; minimale Gewichtung=1)

1) Geringer Platzbedarf der Mechanik => Gewichtung 5
Dieses Kriterium muss im Rahmen der Entscheidung besonders berücksichtigt werden, da das Platzproblem, das größte von allen ist. Daher auch die höchste Gewichtung für dieses Kriterium.

2) Variable Position des Inkrementalgebers beim Einbau => Gewichtung 4
Dieses Kriterium ist aufgrund der teilweise schwer abzuschätzenden Abmaße des Gehäuses relativ zu den Motoren ein weiterer Aufgabenpunkt für die Lösungsvariante. So muss es möglich sein, beim Einbau des Inkrementalgebers und der Mechanik sehr

flexibel zu sein und im oben angegebenen Bereich die Mechanik zu variieren. Die Gewichtung fällt daher fast genauso stark wie Punkt eins aus.

3) Spielfreiheit => Gewichtung 4

Dieses Kriterium ist generell für technische Konstruktion wichtig, da hier die Fehler der Messwerte minimiert werden. Daher Gewichtung ebenfalls relativ hoch.

4) Fertigungsaufwand => Gewichtung 3

Dieses Punkt ist relativ zu den anderen Punkten von nicht ganz so großer Tragweite, da die Mittel für eine Fertigung zu Verfügung stehen und sich die Lösungsvarianten vom Kostenaspekt nicht zu drastisch unterscheiden.

Entscheidung

Die Entscheidung mit Hilfe einer Bewertungstabelle war relativ interessant:

Bewertungstabelle für Nick-Motor

	Prinzip =>	Translatorisch			Rotatorisch (An Potentiometer Kupplung)			Rotatorisch (An Stickmechanik Aufnahme)		
Bewertungskriterien	Gewichtung	Starre Verbindung	Bowdenzug	Zahnstange an Hebelarm	Zahnriehmen	Seilzug	Zahnstange mit Ritzeln	Zahnriehmen	Seilzug	Zahnstange mit Ritzeln
Aus Teil Eins (Prinzipien)										
Aufwand	5	3			4			2		
Mechaniksymmetrie	5	1			5			1		
Platz am Abgriff	4	5			3			2		
Aus Teil Zwei (Lösungsvarianten)										
Platzbedarf der Mechanik	5	3	5	2	3	4	2	3	4	2
Variable Position des Ink. Gebers	4	3	4	3	2	5	5	2	5	5
Spielfreiheit	4	4	3	2	5	3	2	5	3	2
Fertigungsaufwand	3	5	5	2	5	3	4	5	3	4
Gesamt (mit Gewichtung)		19,6	21,6	15,2	23	23,6	21,4	16,2	16,8	14,6

Bewertungstabelle für Roll-Motor

	Prinzip =>	Translatorisch			Rotatorisch (An Potentiometer Kupplung)			Rotatorisch (An Stickmechanik Aufnahme)		
Bewertungskriterien	Gewichtung	Starre Verbindung	Bowdenzug	Zahnstange an Hebelarm	Zahnriehmen	Seilzug	Zahnstange mit Ritzeln	Zahnriehmen	Seilzug	Zahnstange mit Ritzeln
Aus Teil Eins (Prinzipien)										
Aufwand	5	5			4			0		
Mechaniksymmetrie	5	1			5					
Platz am Abgriff	4	5			3					
Aus Teil Zwei (Lösungsvarianten)										
Platzbedarf der Mechanik	5	3	5	2	3	4	2			
Variable Position des Ink. Gebers	4	3	4	3	2	5	5			
Spielfreiheit	4	4	3	2	5	3	2			
Fertigungsaufwand	3	5	5	2	5	3	4			
Gesamt (mit Gewichtung)		21,6	23,6	17,2	23	23,6	21,4			

Tabelle 3.2 : Bewertungstabelle

Für den Nick Motor ist knapper Sieger der Seilzug, kurz vor dem Zahnriemen.
Für den Roll-Motor gibt es ein Unentschieden zwischen dem Seilzug und dem Bowdenzug (zwei Lösungsvarianten, die auf unterschiedlichen Prinzipien beruhen !!)

Die endgültige Entscheidung zum Bau einer Inkrementalgeber Mechanik für beide Achsen, fällt nun auf den Seilzug, da er insgesamt am besten abschneidet und vor allem auch bei beiden Achsen gleich zu realisieren ist. Dadurch verringert sich der Konstruktionsaufwand erheblich. Allerdings müssen alle Teile von der Werkstatt gefertigt werden, wobei sich aber die Anzahl der Teile in Grenzen halten wird.
Diese Lösungsvariante ist somit ein guter Kompromiss zwischen Fertigungsaufwand und Anforderungen an Genauigkeit und Variabilität.

3.2 Realisierung der Inkrementalgeber Mechanik

Der Einbauort

Bei den Untersuchungen zum Einbau der Inkrementalgeber ist das größte Problem der Platzbedarf im Stick. Dies betrifft insbesondere den Rollmotor, da dieser an allen Stellen so eng am Stickgehäuse verschraubt ist, dass es schließlich nur eine Möglichkeit gibt, um den Inkrementalgeber parallel zur Achse platzieren zu können, so dass die Seilzugmechanik realisiert werden kann. Es ist schließlich eine Mechanik aus verschiedenen Abstands-elementen und einer Haupt-halterung realisiert, welche exakt in den geringen Bauraum beim Rollmotor integriert wird. Der Platzbedarf des Nickmotors ist im Verhältnis dazu mehr als ausreichend, so dass die gleiche Mechanik dort ohne Probleme verwendet werden kann.

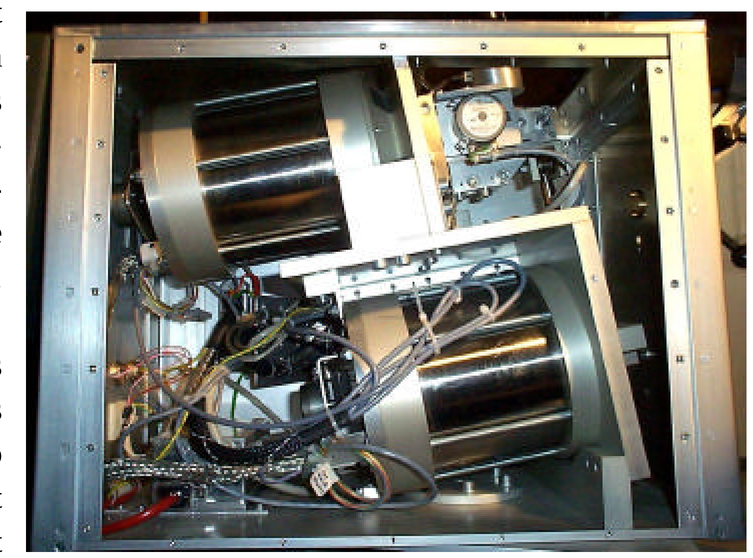


Abbildung 3.1 : Platzangebot im Stickgehäuse

Die Kraft-Kupplung

Um einen Kraftschluss der Inkrementalgeber mit der Motorwelle zu ermöglichen, muss die bestehende Kupplung zum Winkelsensor (Potentiometer) teilweise ersetzt werden um sowohl den Inkrementalgeber als auch weiterhin das Potentiometer an der Welle des Motors betreiben zu können. Die Kupplung besteht dabei aus zwei (wäscheklammerartigen) Teilen (siehe Abbildung 3.2 und Abbildung 3.3) welche eine Kraftübertragung zum Winkelsensor nur in rotatorischer Richtung zulassen, jedoch einen eventuellen Versatz der Wellen tolerieren. Damit wird gewährleistet, dass die Welle des Potentiometers keinen Verspannungen ausgesetzt wird. Bei dem Einbau des Inkrementalgebers muss nun der eine Teil der Wellenkupplung durch das Reibrad des Seilzuges ersetzt werden und gleichzeitig die Aufnahme des anderen Teils der Wellenkupplung bestehen

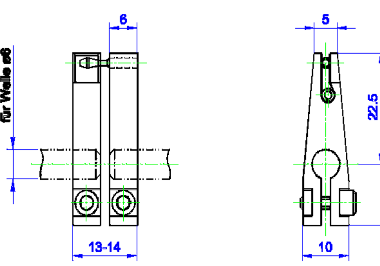


Abbildung 3.2 : Wellenkupplung

bleiben. Dazu ist in dem Reibrad an der Motorwelle die spezielle Aufnahme der Kupplung eingefräst. (siehe technische Zeichnung „Rolle Motorwelle“ im Anhang III.3) Bei dem Einbau ist weiterhin darauf zu achten, dass kein Verstellen des Winkels durch den Umbau verursacht wird, da sonst die Torque Ribble Kompensation nicht mehr korrekt arbeitet. Dazu existiert eine Umbau Checkliste, welche im Anhang III.4 zu finden ist.

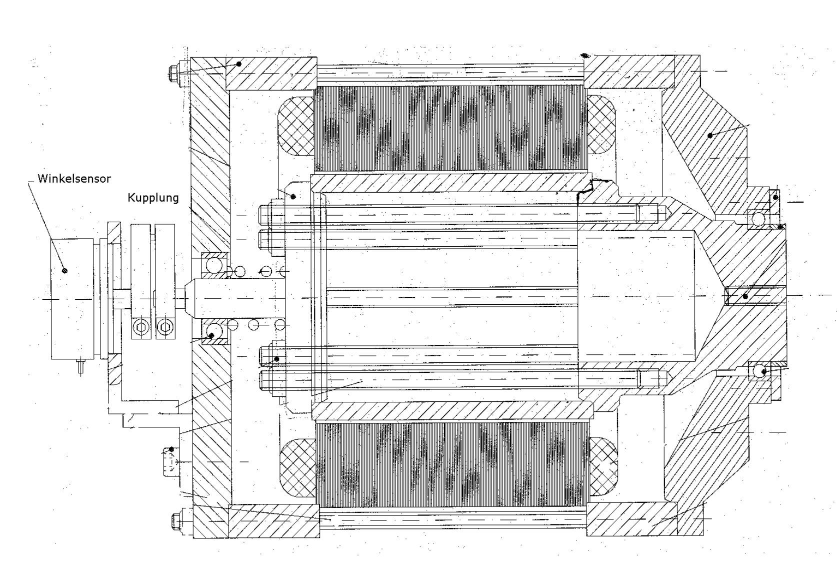


Abbildung 3.3 : Motor mit Kupplung

Montagefreundlichkeit der Mechanik

Aufgrund des sehr geringen Platzangebotes beim Rollmotor ist die Mechanik so konstruiert, dass eine variierbare Montage sowohl zum Abstand des Motors als auch rotatorisch um die Motorachse erfolgen kann. Dies wird durch Langlöcher in den jeweiligen kraftschlüssigen Verbindungen realisiert. Bei der Montage kann so die Inkrementalgebermechanik exakt und verspannungsfrei ausgerichtet werden. (siehe Abbildung 3.4)

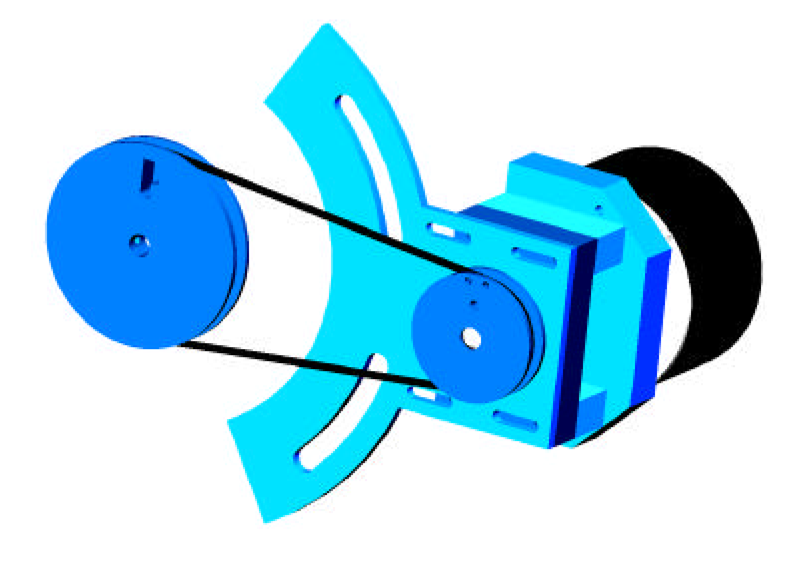


Abbildung 3.4 : Inkrementalgebermechanik

Das Seil

Bei dem Seil handelt es sich um ein extrem dehnungsarmes Seil (mit Trevira© Kern) aus dem Drachensport, welches bei einer möglichen Belastung von 80 kg nur 0.01 % Dehnung aufweist. Für die Anwendung an der Inkrementalgebermechanik, bei der

deutlich geringer Kräfte aufgebracht werden müssen, ist die Dehnung somit vernachlässigbar klein. Der Seilschlupf um die Metallräder, wird durch ein einfaches Umschlingen beider Räder optimiert, so dass selbst bei hohen dynamischen Bewegungen keine Kräfte auftreten, die das Seil zum Durchrutschen bringen können.
Das Fixieren des Seils erfolgt durch das Durchziehen des Seiles durch zwei Löcher und anschließender Klemmverschraubung seitlich an der Rolle der Inkrementalgeberwelle (die kleinere Rolle). Ein endgültiges Spannen des Seils wird durch leichtes Ziehen der Inkrementalgeber Lagerblöcke längs der Langlochverschraubung erreicht.

Das Ergebnis der Montage der Inkrementalgebermechanik wird in den folgenden Abbildungen dargestellt. Dabei zeigen Abbildung 3.5 und Abbildung 3.6 das Copilotenstickgehäuse, welches deutlich mehr Raum für die Montage zulässt, als das Pilotenstickgehäuse, bei dem die Messverstärker im Stickgehäuse untergebracht sind.

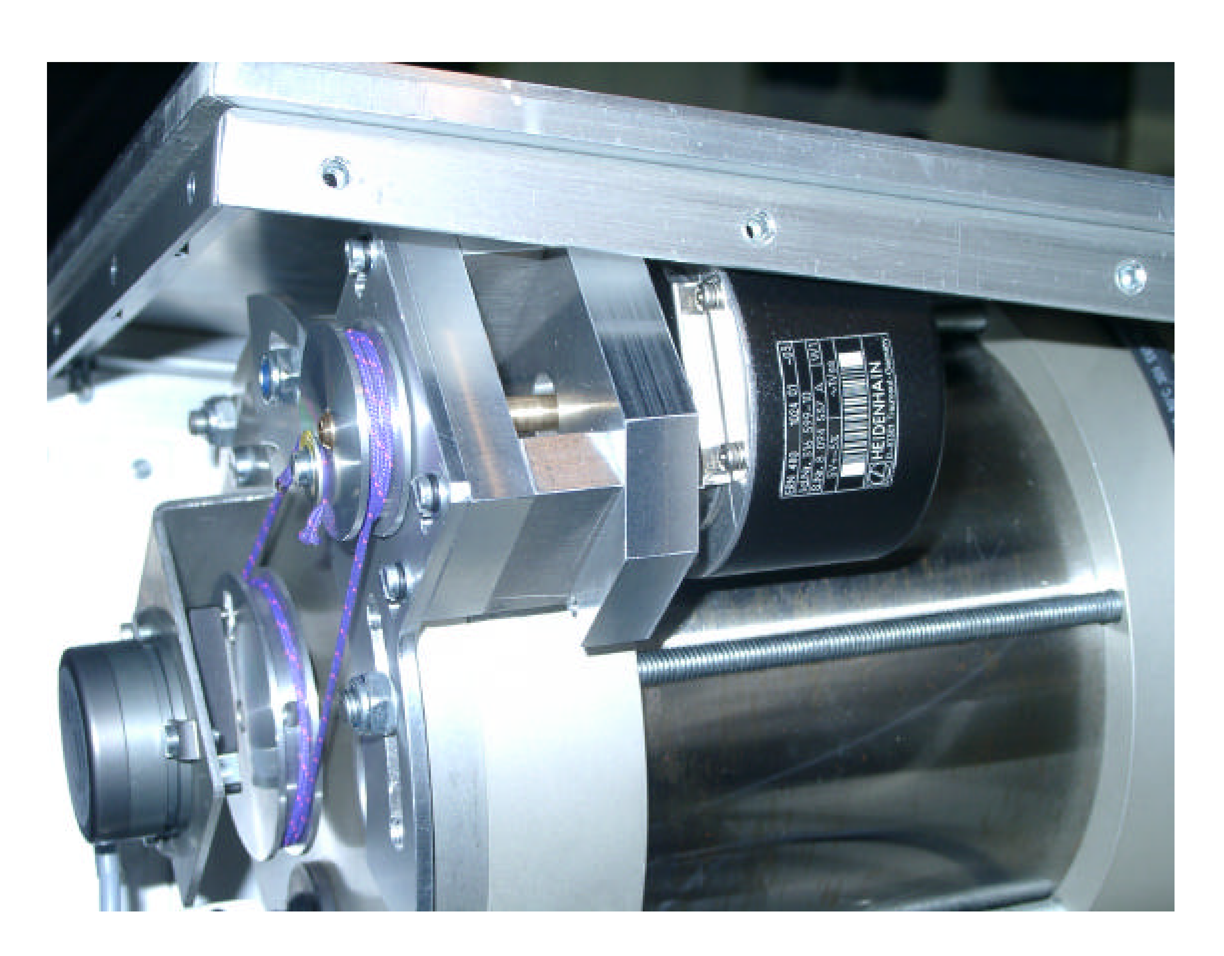


Abbildung 3.5 : Inkrementalgeber Mechanik am Rollmotor (Copilotstickgehäuse)

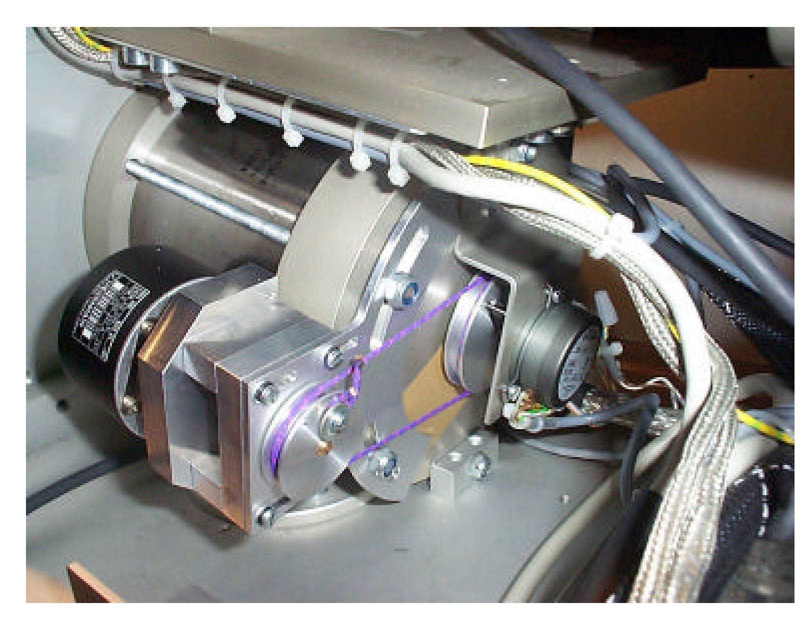
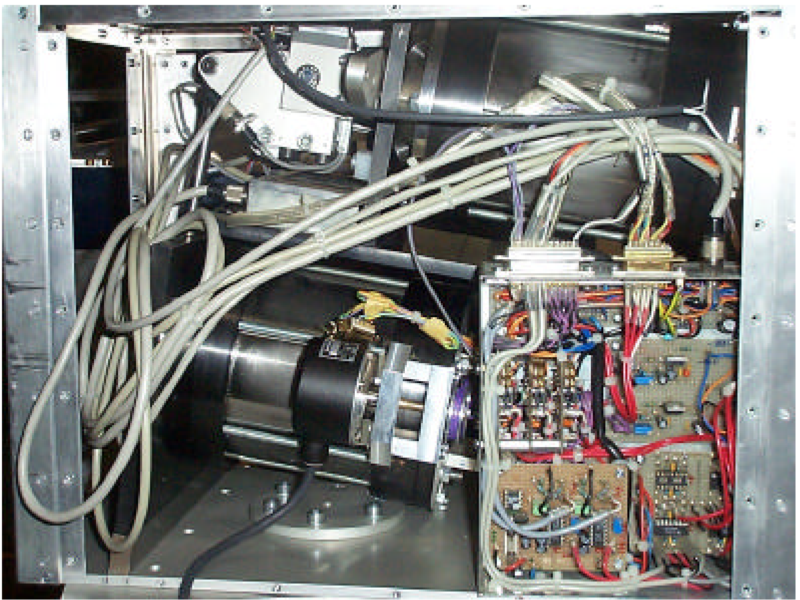


Abbildung 3.6 :
Inkrementalgeber Mechanik am Nickmotor (Copilotstickgehäuse)

Abbildung 3.7 :
Inkrementalgeber Mechanik am Nickmotor (Pilotenstickgehäuse)



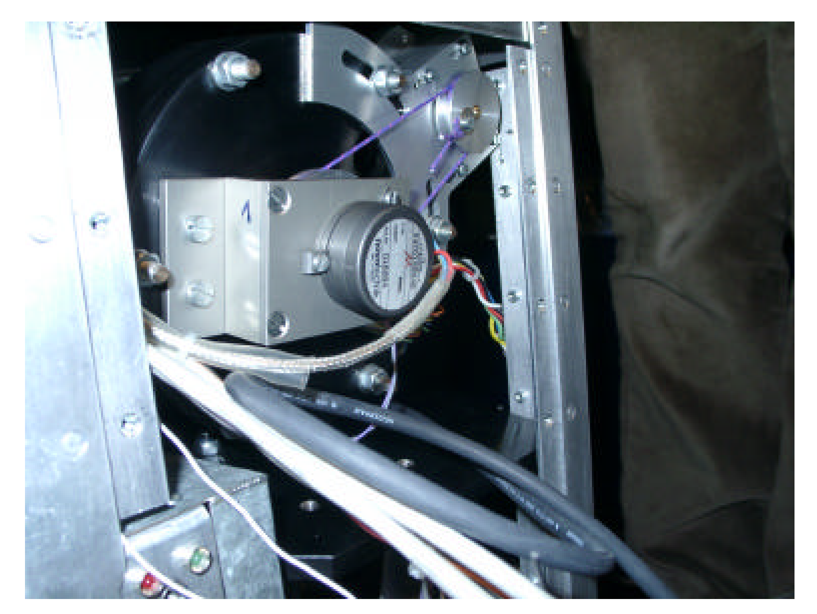


Abbildung 3.8 :
Inkrementalgeber Mechanik am Rollmotor (Pilotenstickgehäuse)

4 Die Software

Die Dokumentation der Software ist in zwei Abschnitte unterteilt:

1) Grundsätzliche Programmierung der Inkrementalgeberkarte für den Betrieb (4.1)
2) Implementierung der Inkrementalgeberfunktionen in die Sticksoftware und Ergänzungen der Sticksoftware (4.1-4.4)

4.1 Programmierung der Inkrementalgeberkarte

Im Folgenden soll nun darauf eingegangen werden, wie die Inkrementalgeberkarte programmiert wird, um Winkel und Winkelgeschwindigkeiten zu erhalten. Alle die erwähnten Prozeduren sind in der Datei *ik121out.c* und der Header Datei *ik121out.h* zusammengefasst. Die dort benutzten Grundfunktionen zur Hardwaresteuerung stehen in der Datei *ik121_0.c* und *ik121_0.c* zur Verfügung.

Die Register:

Alle Datenabrufe und Steuerungen der Inkrementalgeberkarte werden über Register ausgeführt. Die einzelnen Funktionen der Register sind in der Beschreibung der Inkrementalgeberkarte von Heidenhain [4] ausführlich beschrieben, sowie auch die einzelnen Grundfunktionen, mit denen die Register ausgelesen bzw. geschrieben werden können. Hier soll daher nur kurz auf die wichtigsten Funktionen eingegangen werden.

Die wichtigsten Register und Funktionen.

Die Karte verfügt über insgesamt 10 Kontrollregister mit jeweils 16 Bit. Die Adressierung der Register geschieht über die Grundadresse der Karte (base_address) an der ISA Schnittstelle und der Addition einer Adresse für das jeweilig gewünschte Register. Der Schreib/Lesezugriff auf die Register wird durch Prozeduren/Funktionen aus der Datei *ik121_0.c* (bzw. für Pascal *ik121_0.pas*) ermöglicht, welche als Projektfile (bzw. als Unit) zum eigentlichen Quellcode eingefügt werden müssen. In diesen Dateien sind z.B. die Prozedur *write_g26* und die Funktion *read_g26* enthalten, mit denen die Register direkt geschrieben bzw. gelesen werden können. Die Übergabevariablen sind dabei nur die (base_address), die Nr. der Achse (1 oder 2), die Adresse des Registers und (nur bei *write_g26*) der Wert, der geschrieben werden soll. Mit Hilfe dieser Prozeduren/Funktion kann die Karte bereits initialisiert und für den Messwertabruf vorbereitet werden.
Das Abholen der Inkrementalgeber Position geschieht mit der Funktion *read_count_value32* (32 Bit Wert) bzw. *read_count_value48* (48 Bit Wert). Außerdem sei noch die Funktion *pool_latch* zu erwähnen, die vor jedem Lesevorgang ausgeführt werden sollte und die sicher stellt, dass auch ein neuer Wert vollständig in das Datenregister geschrieben wurde.

Initialisierung der Karte:

Alle Aufgaben der Initialisierung werden von der Prozedur *init_ik121* übernommen, welche nur einmal zu Beginn der Messwertaufnahmen aufgerufen werden muss. In der Prozedur wird als erstes in das Initialisierungsregister (0Eh) geschrieben. Hier wird der Interpolationsmode eingeschaltet, so dass die Karte 48 Bitwerte ausliest. Anschließend muss im Kontrollregister das Error Bit gelöscht werden und der Zähler gestartet werden.

Abholen der Messwerte:

Nach dem Initialisieren kann innerhalb einer Schleife die Datenregister immer wieder neu gelesen werden und somit die Werte in einem festen Intervall aktualisiert werden. Dazu wird die Karte über das Kontrollregister 1 (0Eh) angewiesen, den aktuellen Zählerstand in das Datenregister zu schreiben. Mit der Funktion *pool_latch* wird so lange gewartet, bis dies geschehen ist und anschließend der Wert mit *read_count_value48* ausgelesen. Anschließend kann der gewonnene Wert auf dem Bildschirm oder auch in eine Datei geschrieben werden und die Schleife von neuem beginnen.

Eine Übersicht über die beschriebene Initialisierung und Messwert Erfassung soll Abbildung 4.1 geben:

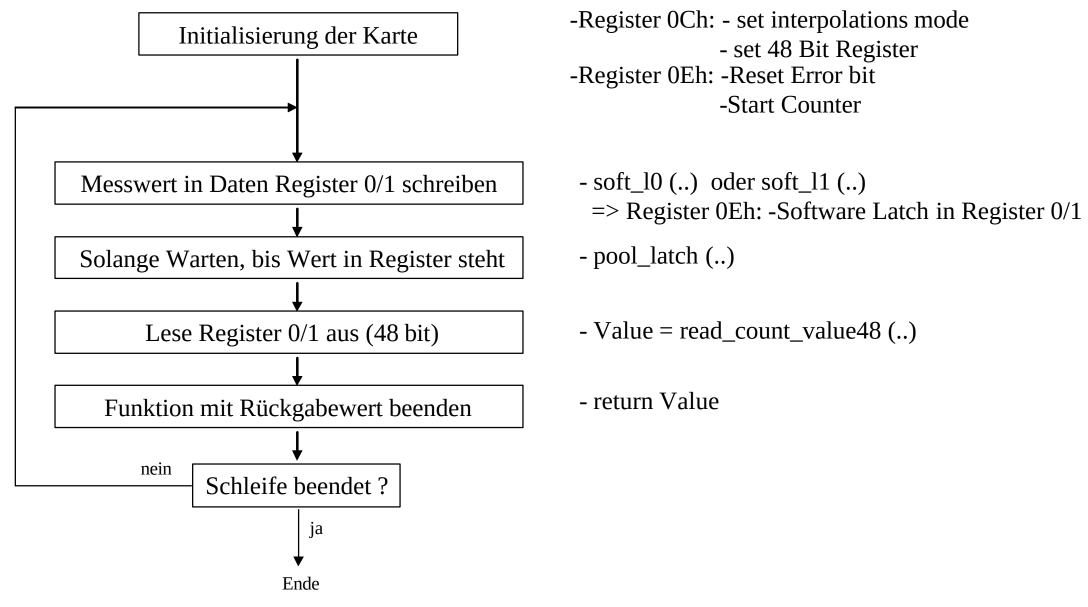


Abbildung 4.1 : Grundalgorithmus zum Abruf der Winkeldaten

Generierung des Geschwindigkeitssignals

Um eine Winkelgeschwindigkeit zu berechnen, müssen zwei aufeinander folgende Winkel gemessen werden, deren Differenz die Winkeländerung pro Zeitabschnitt und damit die diskrete Ableitung ergeben.

$$dy = \frac{(y_2 - y_1)}{T_{Periode}} \qquad (4.1)$$

Entscheidend für die Auswertbarkeit des Signals ist dabei, wie groß die Zeitdifferenz $T_{Periode}$ gewählt wird. Um dies zu entscheiden, werden mit den bereits in den Sticks eingebauten Inkrementalgebern Tests durchgeführt, in denen die Winkelgeschindigkeit während eines realistischen Fluges im Simulator über einen längeren Zeitraum aufgezeichnet wird. Dabei wird $T_{Periode}$ variiert und die Ergebnisse anhand einer Tabelle aufgezeichnet und anschließend in einer Excel Grafik anschaulich dargestellt. (siehe dazu Anhang II.1) (Dokumentation der Testsoftware siehe VI.1)
Die Ergebnisse zeigen, dass Ableitungen mit $T_{Periode}$ > 10 ms einen Betrag annehmen, der für eine Weiterverarbeitung gut geeignet ist. Um so größer jedoch $T_{Periode}$ gewählt wird, um so träger ist auch das Signal was eine ungünstige Verzögerungen zu Folge hat.

Da innerhalb der Sticksoftware der Messwertabruf fest mit einer Periode von 1 ms erfolgt, wird unter Berücksichtigung der Testergebnisse eine Messwertschar als Geschwindigkeitssignal realisiert.
Dazu wird eine Feldvariable mit 10 Variablen definiert. Die ersten 10 Winkeldifferenzen werden dort nacheinander abgespeichert. Der 11 Wert überschreibt dann wieder den ersten, der 12 den zweiten usw. Von der gesamten Feldvariable wird nach jedem Abspeichern die Summe berechnet. (siehe Abbildung 4.2)

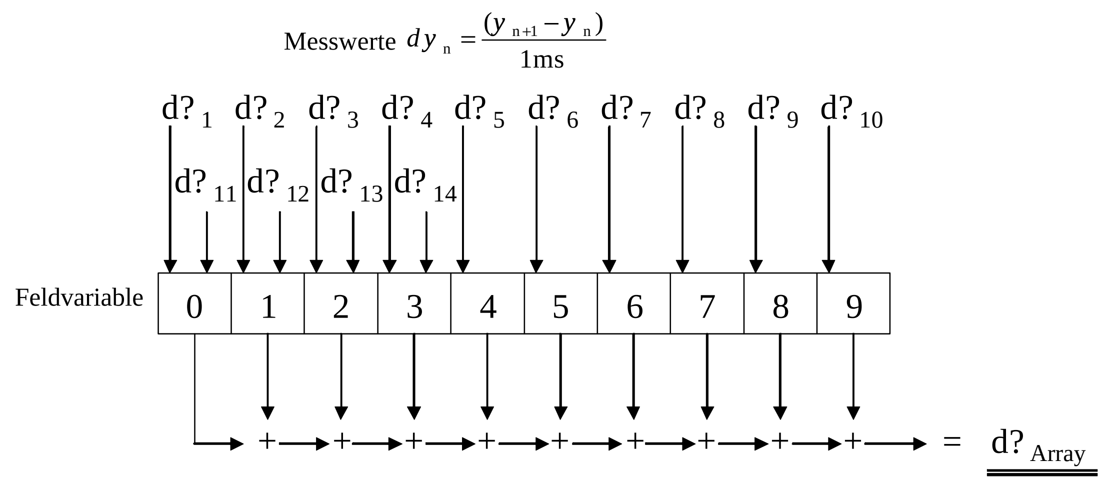


Abbildung 4.2 : Geschwindigkeitssignal Generierung

Der berechnete Wert entspricht dynamisch gesehen näherungsweise einer Periodenzeit von $T_{Periode}$ = 1 ms, vom Betrag her jedoch einer Periodenzeit von $T_{Periode}$ = 10 ms.
Auf diese Art ist das dynamische Verhalten deutlich verbessert, während gleichzeitig ein Signal mit hohem Betrag generiert wird.
In der Praxis hat sich das dynamische Verhalten dieser Anwendung als sehr positiv herausgestellt, wobei in der Konfigurationsdatei *stick.cfg* die Größe der Feldvariable zwischen 1 und 30 variiert werden kann. In der Tabelle I.1 wird die Abhängigkeit zwischen maximaler Dämpfung und $T_{Periode}$ dargestellt, wobei generell die Dämpfung um so höher gewählt werden kann, je größer auch $T_{Periode}$ ist.

4.2 Die Sticksoftware

Die Programmierung der Stick-Hardware und der auswertenden Regelung ist relativ komplex und kann hier nicht in vollem Umfang dargestellt werden. Es soll jedoch eine grobe Übersicht über die Abläufe und die programmierte Regelung gegeben werden, um anschließend die Änderungen, die für die Inkrementalgeber Implementierung und die Kopplung nötig waren nachvollziehen zu können.
Die folgende Abbildung soll einen Überblick über die relevanten Abläufe geben, wobei auch die ausführenden Funktion genannt werden.

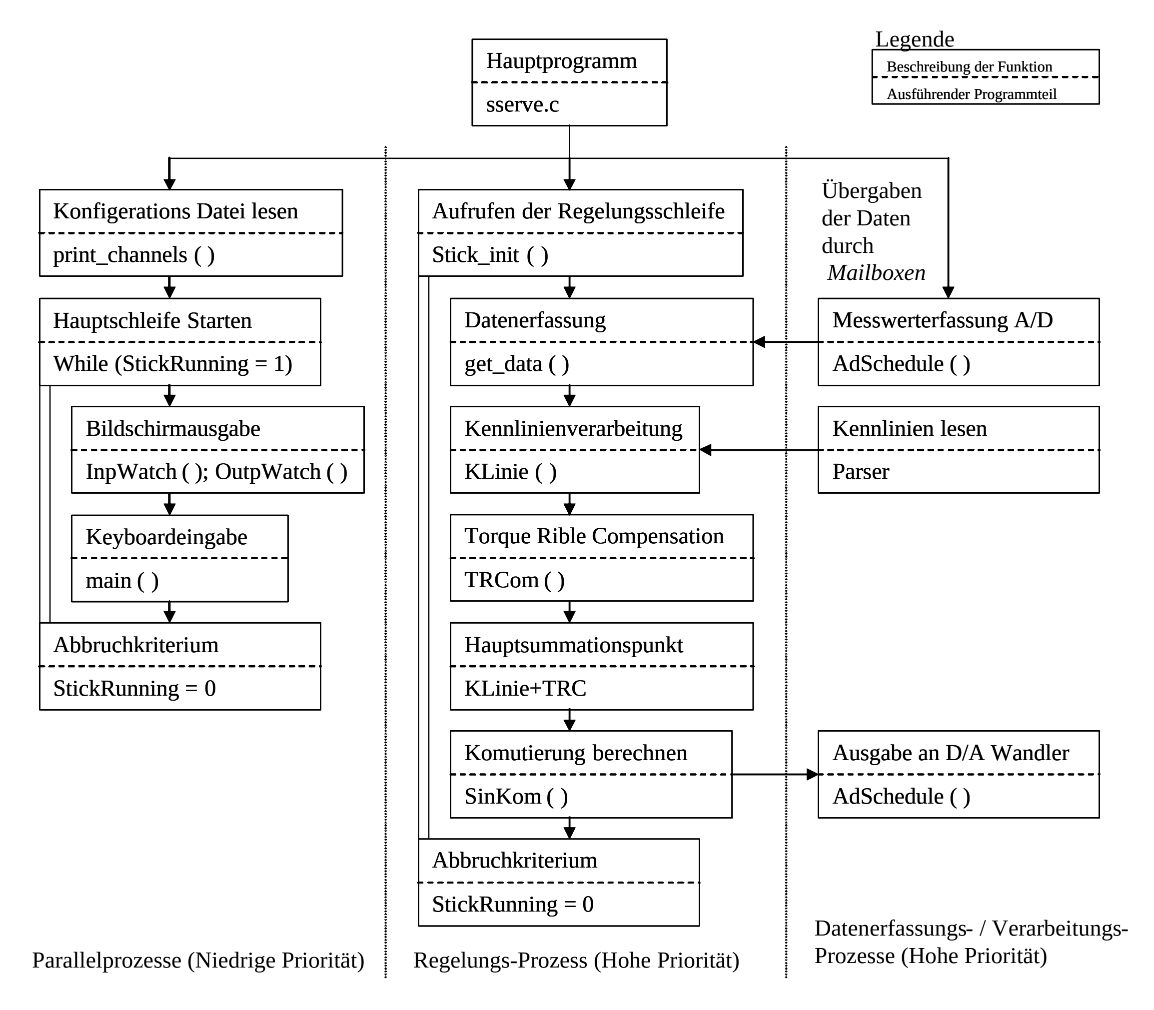


Abbildung 4.3 : Sticksoftware Prozesse

Das Hauptprogramm *sserve.exe* beinhaltet alle Funktionen, die für die Messwertaufnahme, Regelung und Aktorsteuerung nötig sind. Weiterhin ermöglicht es gleichzeitig eine Anzeige aller relevanten Daten auf dem Bildschirm, sowie die Eingabe

von bestimmten Steuerfunktionen durch den Benutzer, wie z.B. eine Änderung der Kennlinie.
Wie in Abbildung 4.3 dargestellt, laufen die einzelnen Prozesse der Regelung, I/O Steuerung und Benutzerschnittstellen zeitlich parallel ab. Die Realtime Kernel Programmierung ermöglicht es dabei durch die Mailboxen, dass die Prozesse untereinander Daten austauschen, ohne jeweils auf die Antwort des anderen Prozesses warten zu müssen. Er stellt dazu sicher, dass Prozesse, wie z.B. die Datenerfassungsschleife mit der Vorgabe von 1ms exakt wiederholt werden, um so die Echtzeitregelung zu ermöglichen. Weitere Dokumentation der Prozesse und Funktionen sind im Anhang IV und VII zu finden. Vorallem die Frage, welche Funktionen in welchen der zahlreichen Quellcode Dateien implementiert sind, wird dort erläutert.

Nun stellt sich die Frage, in welche Prozesse eingegriffen werden muss, um eine Dämpfung mit dem Inkrementalgebersignal realisieren zu können.

4.3 Erweiterung der Sticksoftware (Konzept)

4.3.1 Implementierung des Inkrementalgebers

Um den Betrieb des Inkrementalgebers und die Verarbeitung der Messwerte innerhalb der Sticksoftware zu ermöglichen, müssen folgende Vorraussetzungen geschaffen werden:

1) Das Abrufen der Inkrementalgebermesswerte muss innerhalb eines definierten Intervalls möglich sein, damit eine Echtzeitverarbeitung gewährleistet werden kann.
Dazu müssen zusätzliche Treiber für die Inkrementalgeberkarte geschrieben werden, welche die Datenübergabe an die Sticksoftware ermöglichen.
Weiterhin müssen Funktionen zur Berechnung der Geschwindigkeit programmiert werden, die in der Sticksoftware aufgerufen werden können.

2) Um eine zentrale Eingabe der Skalierwerte und Normierwerte für den Inkrementalgeber zu realisieren ohne den Quellcode neu kompilieren zu müssen, muss eine Funktion zum Abrufen der Werte aus einer Konfigurationsdatei realisiert werden. Dabei ist es nahe liegend, die bisherige Konfigurationsdatei (*stick.cfg*) mit der entsprechenden Auslesefunktion zu erweitern.

3) Das Dämpfungssignal des Inkrementalgebers muss am zentrale Summationspunkt der Regelung aufsummiert werden (siehe Abbildung 2.4).

4.3.2 Implementierung der Kopplung

Um die Ergebnisse der Problemanalyse (Abschnitt 2.4) in ein Konzept zu verwirklichen, müssen folgende Punkte ermöglicht werden:

1) Das Drehmomentsignal des gegenüber liegenden Sticks muss in der Software als Variable vorliegen.

2) Skalierwerte und Normierwerte müssen ebenso wie die Werte der Inkrementalgeber in einer zentralen Datei veränderbar sein.

3) Das Koppelsignal muss am zentralen Summationspunkt der Regelung aufsummiert werden (siehe Abbildung 2.12).

Diese hier aufgestellten Konzepte müssen nun in der bestehenden Software umgesetzt werden.

4.4 Realisierung des Softwarekonzepts

Die folgenden Ausführungen schildern eine Übersicht über die erforderlichen Veränderung der Sticksoftware, ohne jedoch die Software explizit zu beschreiben. Im Anhang befindet sich unter VII eine sehr ausführliche Dokumentation zu jeder Änderung, die in der Sticksoftware während der Studienarbeit vorgenommen wurde, und ist speziell für die Personen gedacht, welche sich mit dem Sticksystem genau vertraut machen müssen.

4.4.1 Veränderungen zur Implementierung der Inkrementalgebersignale

Einlesen der Skalierwerte aus *Stick.cfg*

Das Einlesen der Daten aus der Datei Stick.cfg wird durch die Funktion *print_channels(..)* realisiert. Durch die Ergänzung dieser Funktion können auch die Variablen für die Inkrementalgeber Skalierung, Dämpfung und weitere Daten aus der Konfigurationsdatei eingelesen werden.

Messwertabruf
Die in Abschnitt 4.1 beschriebenen Funktionen *init_ik121 (..)*, *getvalue_ik121 (..)* und *arrayspeed_ik121 (..)* ermöglichen den Messwertabruf innerhalb der Sticksoftware.

Das Initialisieren der Inkrementalgeberkarte mit *init_ik121 (..)* erfolgt dabei beim Starten des Programms *sserve.exe* innerhalb der Funktion *Stick_init (..)*.
Um anschließend zu gewährleisten, dass die Inkrementalgeber Werte genau wie die A/D Wandler Werte mit einer Periode von 1ms aktualisiert werden, wird in der Funktion *get_data (..)* die Funktion *getvalue_ik121 (..)* mit in Regelungsschleife eingefügt. Die Berechnung des Geschwindigkeitssignals wird ebenfalls in der Schleife direkt danach ausgeführt. Damit stehen Winkel und Geschwindigkeitswerte für weiter Berechnungen zur Verfügung.

<u>Hauptsummationspunkt</u>

Die berechneten Geschwindigkeitssignale der jeweiligen Achsen werden anschließend in den Funktionen *RollStick (..)* und *NickStick (..)* am Hauptsummationspunkt der Stickregelung aufaddiert.

4.4.2 Veränderung der Sticksoftware für die Kopplung

Das Drehmomentsignal des gegenüberliegenden Sticks muss vor der Softwareimplementierung erst von der Hardware zu Verfügung gestellt werden. Für die bisherige Kopplung ist jedoch das Winkelsignal des gegenüber liegenden Stick bereits implementiert, wird jedoch für die Drehmomentkopplung nicht mehr benötigt. Daher wird einfach das Winkelsignal durch das Drehmomentsignal ersetzt (Stecker von der Werkstadt umgelötet) so dass die Drehmomentsignale in der Software unter *NickData.passiv* und *RollData.passiv* zur Verfügung stehen.

Für die Kopplung werden die Funktionen *Koppelnick (..)* und *Koppelroll (..)* geschrieben, welche direkt am Hauptsummationspunkt in den Funktionen *RollStick (..)* und *NickStick (..)* aufgerufen werden. Die programmierten Funktionen, welche in dem Quellcode *Koppel.c* eingebunden sind, ermöglichen das Eliminieren des Offsets der Drehmomentsensoren durch die Angabe des Offsets in der Datei *Stick.cfg*. Weiterhin ist es möglich, die Gewichtung der Kopplung in Relation zu den andern Regelungsgrößen einzustellen. Als letztes ist es auch noch möglich, bestimmte Kennlinien der Kopplung zu realisieren, welche ebenfalls durch die *Stick.cfg* Datei ausgewählt werden können.

Die Veränderungen der Sticksoftware können nun auch noch in der Abbildung 4.3 ergänzt werden , so dass sich die Abbildung 4.4. ergibt. Die Fett markierten Teile sollen dabei die veränderten Funktionen bzw. ergänzten Funktionen darstellen.

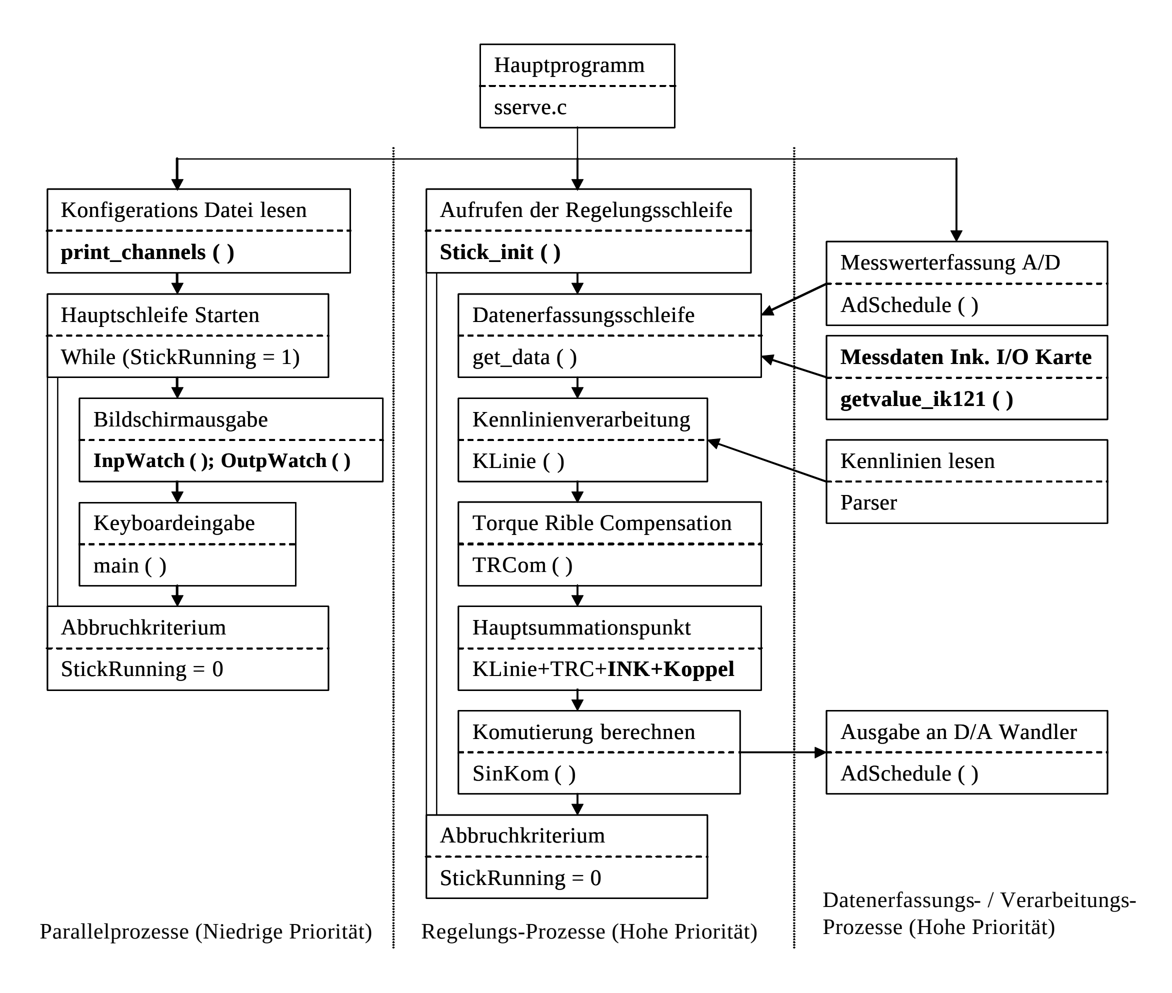


Abbildung 4.4 : Sticksoftware Prozesse (mit INK und Koppel Ergänzung)

(INK = Geschwindigkeitssignal, welches durch die Inkrementalgeber gemessen wird)
(Koppel = Kopplungssignal, welches aus dem Drehmomentsignal des gegenüberliegenden Sticks resultiert)

5 Kritik und Ausblick

Innerhalb der Arbeit konnten alle Teile der Aufgabenstellung mit Erfolg bearbeitet werden. Es entstanden dazu wichtige Grundlagen zur Beschreibung des Sticksystems, die auch für weitere Untersuchungen der Stickregelung wichtig sind.

Durch den Einbau der Inkrementalgeber stehen nun hochpräzise Messsensoren zur Verfügung, welche durch eine entsprechende Erweiterung der Sticksoftware auch die Aufgaben der Potentiometer-Winkelsensoren übernehmen können.

Die Vermeidung von Instabilitäten durch die programmierte Dämpfung ist gewährleistet, so dass nun nahezu alle Kennlinienkonfigurationen verfahren werden können, ohne das eine der untersuchten Instabilitäten auftritt.

Durch das Realisieren der aktiven Dämpfung sind nun ganz neue haptische Eindrücke vermittelbar, die in der Zukunft weiter getestet werden können. Dazu wäre es denkbar, die Dämpfung auch in die Kennliniensprache zu implementieren, so dass auch eine flugzustandsabhängige Dämpfung realisiert werden könnte.

Die realisierte Kopplung ist leider nicht im vollen Funktionsumfang nutzbar, da eine weitere Instabilität des Systems bestimmte Kopplungszustände nicht zulässt. Die auftretende deutlich höherfrequente Instabilität (vorallem beim Copilotenstick) ist sehr wahrscheinlich auf die generelle Problematik der digitalen Regelung zurück zu führen und muss daher in weiteren Arbeiten genauer untersucht werden. Das Konzept der Kopplung ist dabei aber voll funktionsfähig und eine Kopplung der Sticks ist auch bis zu einem bestimmten Koppelgrad möglich.

Mit der Studienarbeit wurde weiterhin versucht, eine Dokumentation an zu fertigen, die die Teile der Sticksoftware und deren Funktionsweise genauer dokumentiert, um ein Einarbeiten in das Sticksystem für künftige Studien- und Diplomarbeiten deutlich zu erleichtern.

6 Literaturverzeichnis

[1] Gehler, Thomas:
Schnelle und sichere Datenübertragung in der Echtzeitumgebung
eines aktiven Steuerknüppels (Diplomarbeit)
TU-Darmstadt, Institut für Flugmechanik und Regelungstechnik, 1997

[2] Steiner, Alexander:
Modellbildung eines aktiven Steuerknüppels (Studienarbeit)
TU-Darmstadt, Institut für Flugmechanik und Regelungstechnik, 1995

[3] Hasse, Karl:
Skript zur Vorlesung „Einführung in die Regelungstechnik“
TU-Darmstadt, Institut für Stromrichtertechnik und Antriebsregelung, 1996

[4] Heidenhain GmbH:
Benutzerhandbuch Ik121 PC Zählerkarte
83301 Traunreut, 7/98

[5] Maccon GmbH
Prospekt „Motoren und Antriebe“
München, 11/92

[6] On Time Informatik GmbH
Dokumentation zu RTK-Kernel V4.0 und V4.5
Eingeheftet in Dokumentation zu Sticksoftware
TU-Darmstadt, Institut für Flugmechanik und Regelungstechnik, 1994

Anhang

I Zusammenfassung Regelungstechnik 2

I.1 Legende zum Blockschaltbild (mit Dämpfungstabelle) 3

I.2 Berechnungen zum Regelungskreislauf 5

I.3 Simulink Ausdrucke 7

II Testergebnisse 9

II.1 Ausdrucke der Excel Graphen 9

III Mechanik 11

III.1 Berechnungen zur Inkrementalgebermechanik 11

III.2 Stückliste 12

III.3 Technische Zeichnungen 13

III.4 Umbau Stick Checkliste 21

IV Der Umgang mit dem Sticksystem 22

IV.1 Ausführbare Dateien 22

IV.2 Konfigerationsdatei Stick.cfg 23

IV.3 Einstellbare Dämpfungswerte 25

IV.4 Speicherort des Quellcodes auf der Festplatte 26

IV.5 Wichtige Programmteile der Sticksoftware 29

IV.6 Programmteile der Inkrementalgeber 29

IV.7 Kompilieren des Codes in .lib / .exe 30

IV.8 Unterschiede der Sticksysteme 31

V Kurzbeschreibung der Erweiterungen 32

VI Inkrementalgeber Programmierung 35

VI.1 Testsoftware Beschreibung 35

VI.2 Flussdiagramme zur Testsoftware 38

VII Sticksoftware Veränderungen im Detail 41

VII.1 Veränderungen zur Implementierung der Inkrementalgeber Software 41

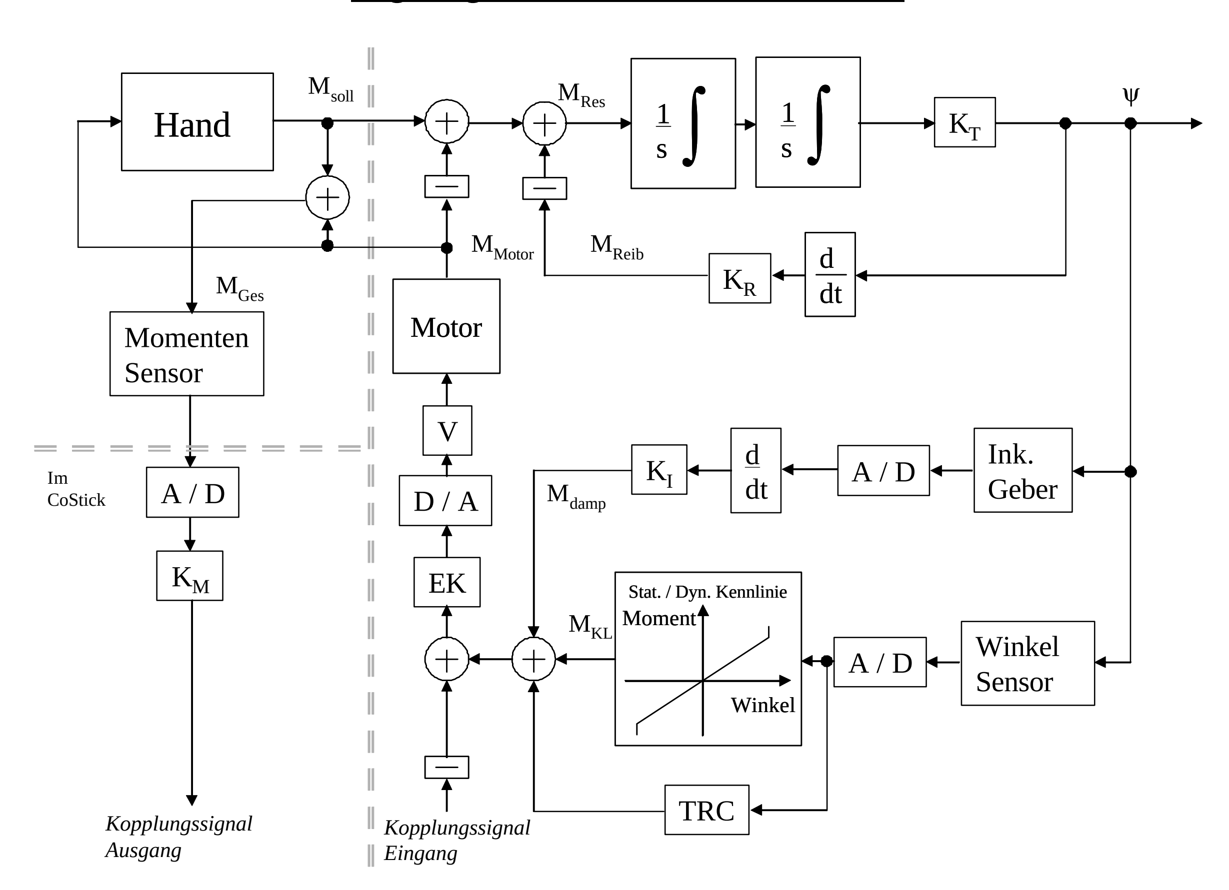
VII.2 Veränderungen der Sticksoftware zur Kopplung der Sticks 46

VIII Softwarecode Testprogramme 49

IX Softwarecode Inkrementalgeber Implementierung 57

I Zusammenfassung Regelungstechnik

Blockschaltbild mit Legende



I.1 Legende zum Blockschaltbild (mit Dämpfungstabelle)

Größe	Wert	Beschreibung des Blockes	Eingang	Ausgang
K_T 1)	R 1 / 0,0125 kgm^2 N 1 / 0,0115 kgm^2	Kehrwert des Trägheitsmoments von Motor + Mechanik + Stick	$\iint M\, d^2t$	ψ $[rad]_A$
K_R 3)	ca. 0,6 Nms/rad (Stick abhängig)	Reibung der Mechanischen Komponenten	ω $[°/s]_A$	M $[Nm]_A$
Winkel Sensor 3)	0,5 V/° $8{,}7 \cdot 10^{-3}$ V/rad	Winkelsensoren (Potentiometer) mit Analogen Verstärkern	ψ $[rad]_A$	U_ψ $[V]_A$
K_L 2)	Variabel (0-5) Nm/°	Kennliniensteigung	ψ $[°]_D$	M $[Nm]_D$
TRC 2)	Dynamisch	Torque Ribble Kompensation	ψ $[°]_D$	M $[Nm]_D$
EK 2)	ca. 0,2 V/Nm	Elektronische Kommutierung des Motors	M $[Nm]_D$	U_s $[V]_D$
V 1)	0,67 A/V	Verstärkung der Servoverstärker	U_s $[V]_A$	I $[A]_A$
Motor 1)	2,35 Nm/A	Elektronische Kommutierter Torque Motor	I $[A]_A$	M $[Nm]_A$
Ink. Geber	2913 Inkremente/° 50,84 Inkremente/rad	Inkremental Geber	ψ $[rad]_A$	U_ψ $[V]_A$
K_I 3)	R 0,00059 * D Nms/° N 0,00049 * D Nms/°	Scallierwerte (konstant) * Dämpfungswerte (variabel) des Inkrementalgebers (zu D siehe Tabelle I.1)	ψ $[°]_D$	M $[Nm]_D$
Momenten Sensor 3)	1 V/Nm	Dehnmessstreifen (DMS) mit Analogen Verstärkern	M $[Nm]_A$	U_M $[V]_A$
K_M	Variabel (0,1-1) Nm/V	Drehmomentscallierung	U_M $[V]_D$	M $[Nm]_D$

Zeitkonstanten

T_V [2)]	1 ms	D / A Wandler zu Servoverstärkern
T_ψ [2)]	1 ms	A / D Wandler des Winkelsignals (der Potentiometer)
T_I [2)]	1 ms	A / D Wandler des Inkrementgebers
T_M [2)]	1 ms	A / D Wandler des Drehmomentsignals (DMS)
T_{array} [2)]	0-30 ms	Software generiertes Zeitintervall für Ableitung des Inkrementalgebersignals

[1)] Quelle: Studienarbeit von Alexander Steiner
[2)] durch Software definiert
[3)] eigene Messungen oder Berechnungen
Abkürzungen: R => Rollachse, N => Nickachse
Zeichenerklärung:
$[Nm]_D$ ist ein Drehmoment, welches digital erfasst wurde.
$[Nm]_A$ ist ein Drehmoment, welches analog bzw. Mechanisch vorliegt.

Maximale Dämpfungswerte für Inkrementalgeber

Die derzeitige Problematik der hochfrequenten Schwingungen (siehe 6) führt dazu, dass bestimmte Dämpfungswerte nicht überschritten werden dürfen. Folgende Tabelle gibt diese Maximas in Abhängigkeit des softwaregenerierten Zeitintervalls der Ableitung an Die angegebenen Minimas geben eine notwendige Dämpfung bei einer eingestellten Kennlinie von Fe 2.0 an, welche aber je nach Kennlinie variieren können.

	Nickachse Dämpfung		Rollachse Dämpfung	
T_{array}	max	min	max	Min
30	5	0,3	15	0,1
20	10	0,3	20	0,1
10	20	0,2	50	0,05
5	20	0,2	68	0,05
1	18	0,2	13	0,05

Tabelle I.1.: Max. Dämpfungswerte für Inkrementalgeber

I.2 Berechnungen zum Regelungskreislauf

<u>Ohne Inkrementalgeber:</u>

Obere Kreisschaltung G1:

$$\frac{1}{sK_R + s^2 \frac{1}{K_T}}$$

Unterer Parallelschaltung G2

$$K_A \cdot K_L$$

Gesamtes Übertragungsverhalten ohne Inkrementalgeber (Kreisschaltung)

$$\frac{1}{K_A K_L + sK_R + s^2 \frac{1}{K_T}} = \frac{\frac{1}{K_A K_L}}{1 + s\frac{K_R}{K_A K_L} + s^2 \frac{1}{K_T K_A K_L}}$$

Dämpfung und Schwingungsfrequenz

Dämpfung allgemein bei PT_2

$$\frac{K}{1 + sT_0 + s^2 T_0 T_1} \qquad d = \frac{1}{2}\sqrt{\frac{T_0}{T_1}} \qquad \nu_0 = \frac{1}{\sqrt{T_0 T_1}}$$

⇨ Für Stickregelung

$$T_0 = \frac{K_R}{K_A K_L} \qquad T_1 = \frac{1}{K_R K_T}$$

⇨ Dämpfung ohne Inkrementalgeber Kreisfrequenz

$$d = \frac{1}{2}\sqrt{\frac{K_T K_R^2}{K_A K_L}} \qquad \nu_0 = \sqrt{K_T K_A K_L}$$

Mit Inkrementalgeber

Obere Kreisschaltung G1:

$$\frac{1}{sK_R + s^2 \frac{1}{K_T}}$$

Untere Parallelschaltung G2

$$K_A \cdot (K_L + sK_D)$$

Gesamtes Übertragungsverhalten mit Inkrementalgeber (Kreisschaltung)

$$\frac{1}{K_A K_L + sK_A K_D + sK_R + s^2 \frac{1}{K_T}} = \frac{\frac{1}{K_A K_L}}{1 + s\frac{(K_R + K_A K_D)}{K_A K_L} + s^2 \frac{1}{K_T K_A K_L}}$$

Dämpfung allgemein bei PT_2

$$\frac{1}{1 + sT_0 + s^2 T_0 T_1} \qquad d = \frac{1}{2}\sqrt{\frac{T_0}{T_1}} \qquad v_0 = \frac{1}{\sqrt{T_0 T_1}}$$

⇨ Für Stickregler

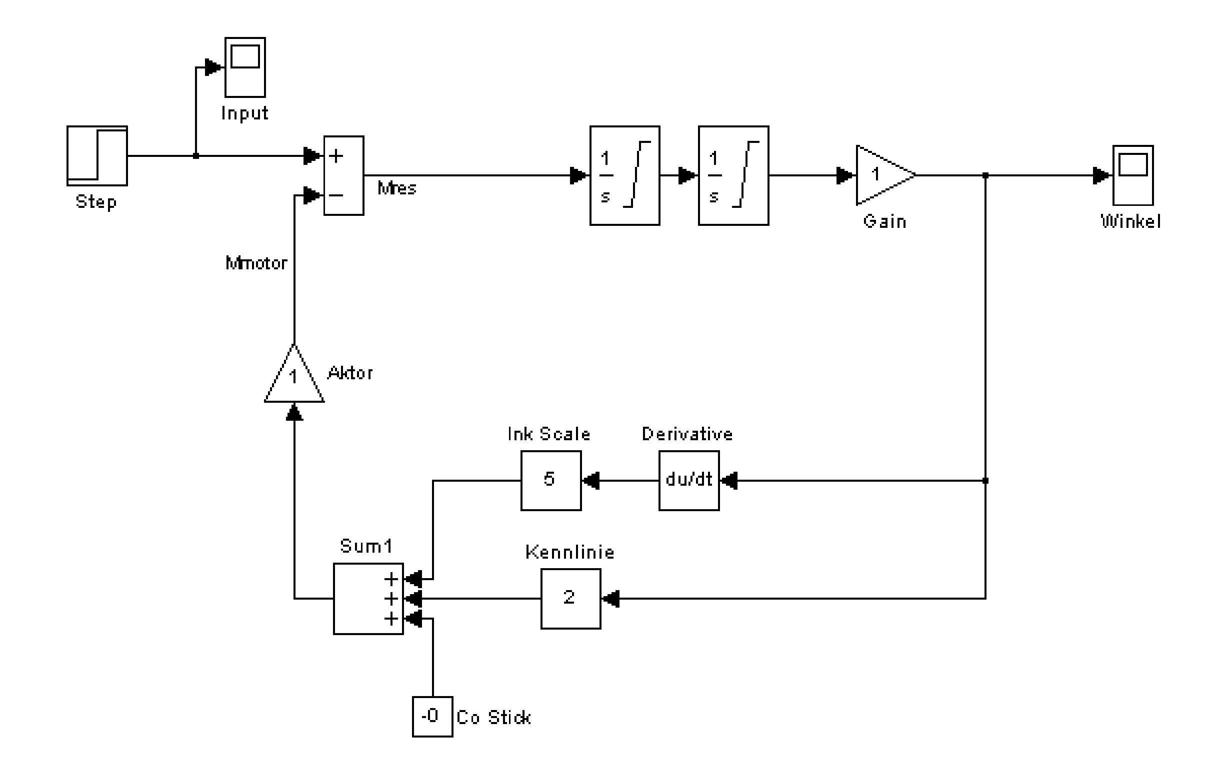
$$T_0 = \frac{K_R + K_A K_D}{K_A K_L} \qquad T_1 = \frac{1}{K_T (K_R + K_A K_D)}$$

⇨ Dämpfung mit Inkrementalgeber Kreisfrequenz

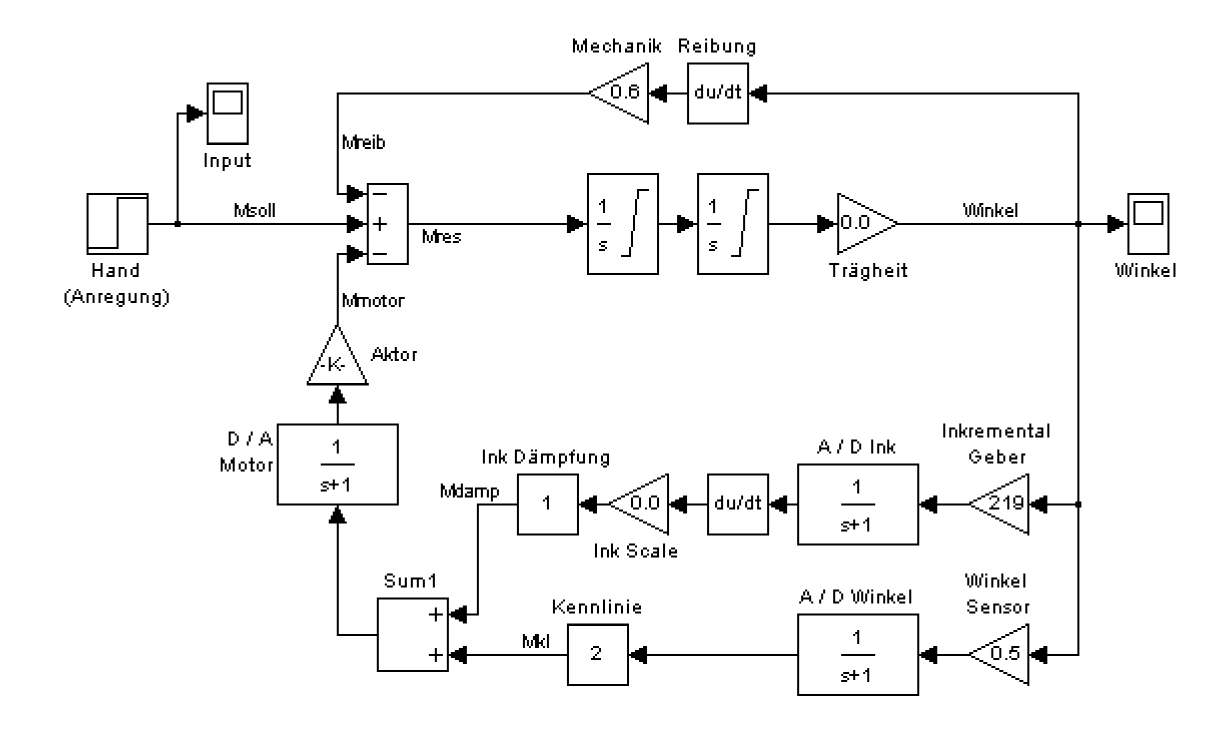
$$d = \frac{1}{2}\sqrt{\frac{K_T (K_R + K_A K_D)^2}{K_A K_L}} \qquad v_0 = \sqrt{K_T K_A K_L}$$

I.3 Simulink Ausdrucke

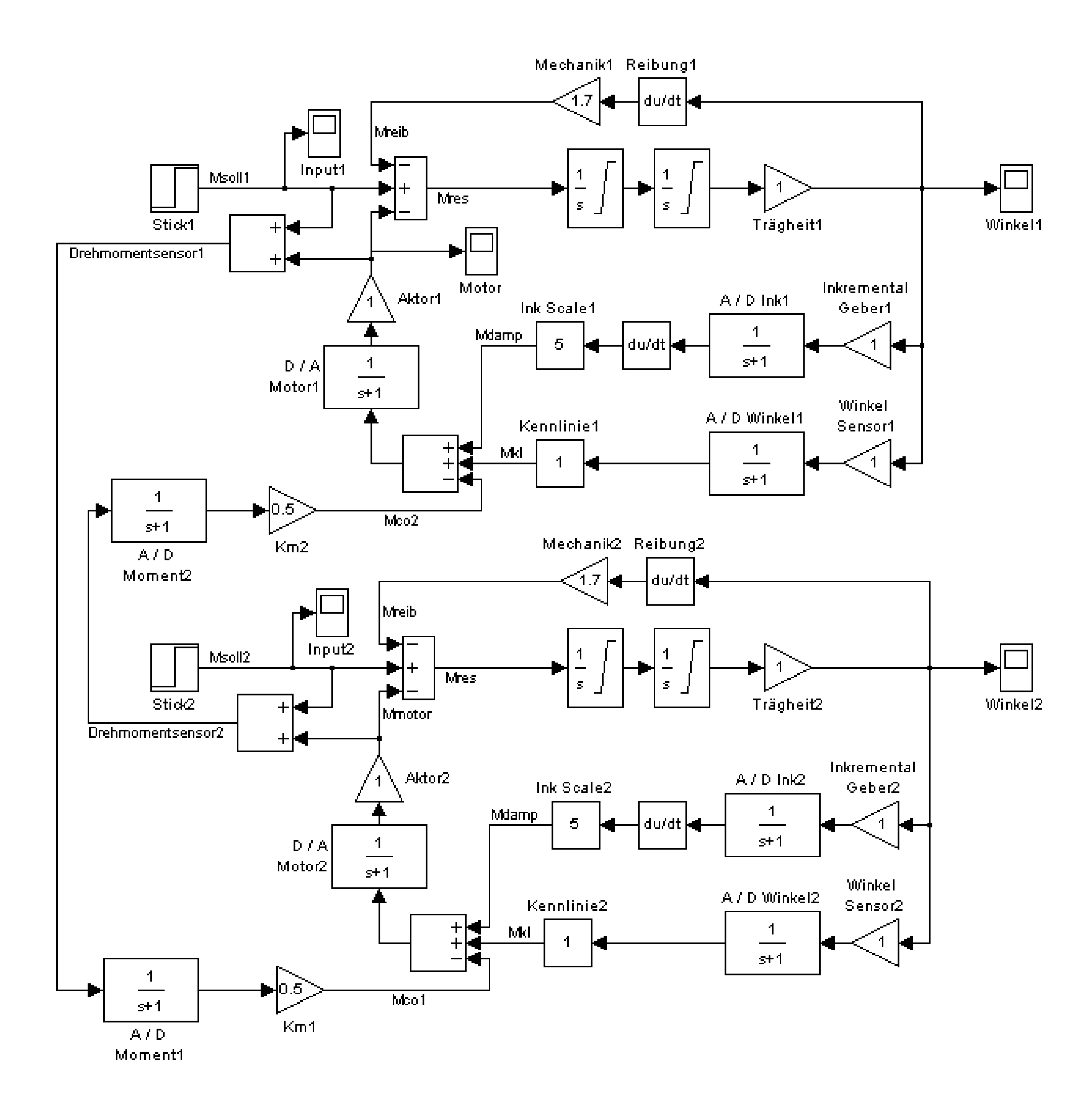
Stickmodell einfach (ohne Verzögerungsglieder



Stickmodell einfach (mit Verzögerungsglieder)



Stickmodell der Drehmomentkopplung

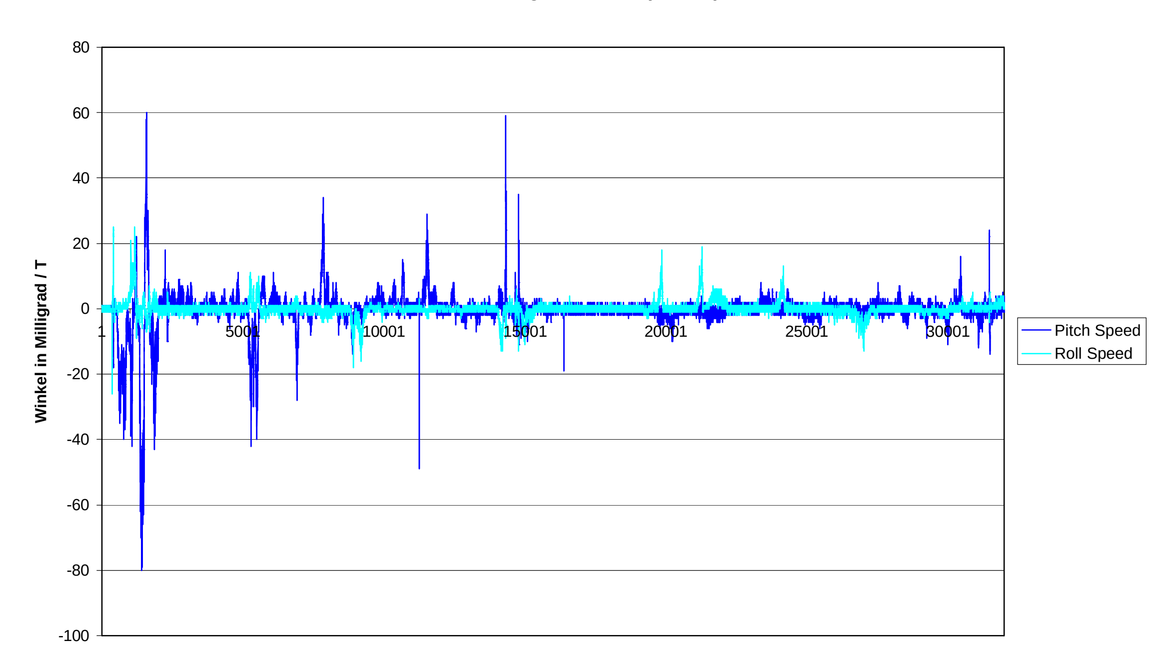


II Testergebnisse

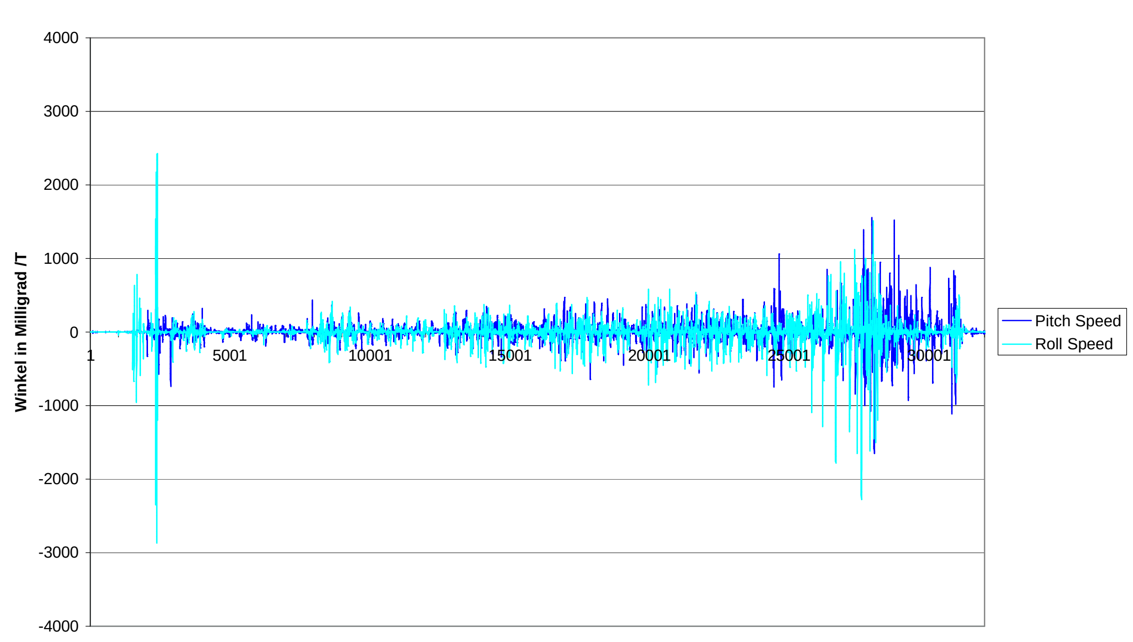
II.1 Ausdrucke der Excel Graphen

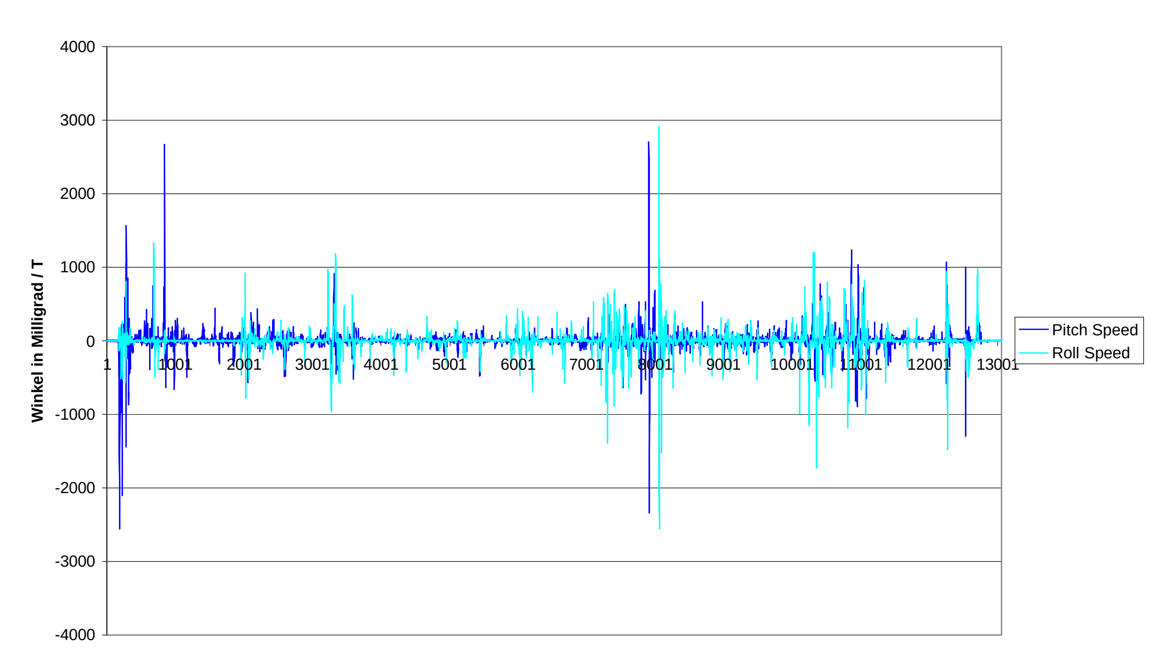
In den folgenden Graphen sind die Geschwindigkeitssignale der Inkrementalgeber bei einem Testflug festgehalten. Damit soll festgestellt werden, welche Amplitude bei welcher Abtastfrequenz zu erwarten sind, so dass ein Eindruck davon gewonnen werden kann, ob die Signale überhaupt auswertbar sind.
Im Folgenden ist darauf zu achten, wie groß die Amplituden in den unterschiedlichen Graphen sind, wobei die Y-Achse auch jeweils unterschiedlich skaliert ist !

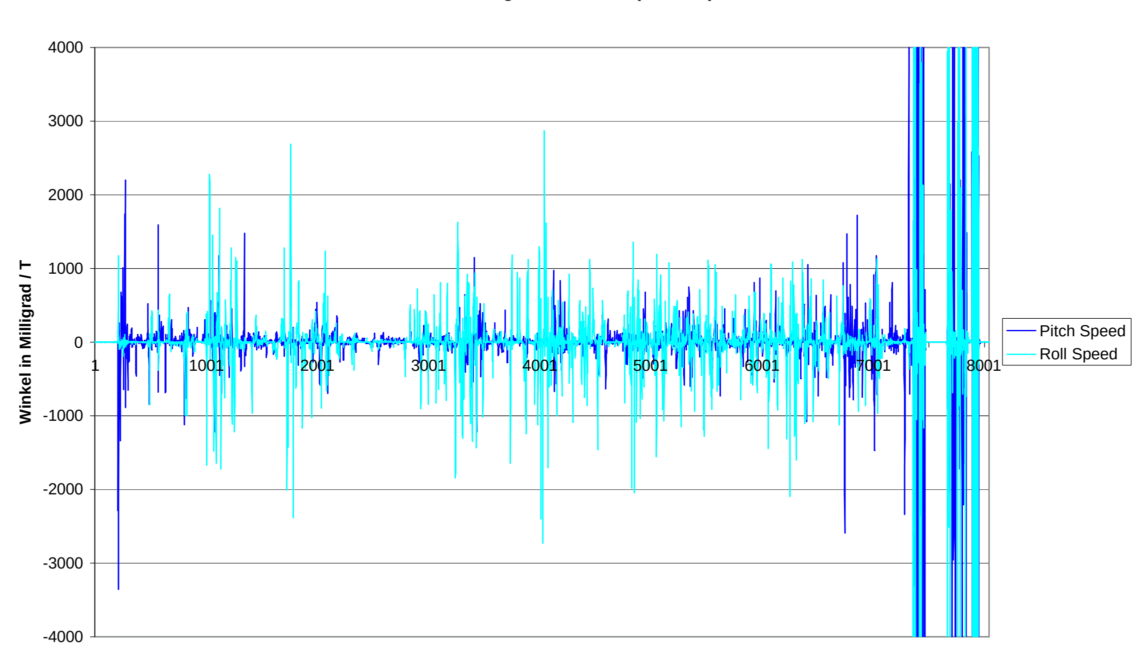




Abtastung mit T=10 ms (f=100 Hz)




Abtastung mit T=20 ms (f=50 Hz)
Winkel in Milligrad / T
4000
3000
2000
1000
0
-1000
-2000
-3000
-4000
1
1001
2001
3001
4001
5001
6001
7001
8001
9001
10001
11001
12001
13001
Pitch Speed
Roll Speed


Abtastung mit T=33 ms (f=30 Hz)
Winkel in Milligrad / T
4000
3000
2000
1000
0
-1000
-2000
-3000
-4000
1
1001
2001
3001
4001
5001
6001
7001
8001
Pitch Speed
Roll Speed

III Mechanik

III.1 Berechnungen zur Inkrementalgebermechanik

Auflösung der Inkrementalgeber

Inkrementalgeber Striche: 1024

I / O Karte Interpolation 1024fach

Bei einer Umdrehung, ergeben sich: 1.048.576 Inkremente. (Interpoliert)

Auflösung = 1.048.576 / 360 = 2912 Inkremente / grad (Interpoliert)

Allerdings ist bei dieser Interpolation die letzte Stelle sehr ungenau, weshalb für die Geschwindigkeitsmessung nur eine Auswertung der interpolierten Inkremente ab

Der zweiten Stelle sinnvoll ist.

Daher ergibt sich eine **Auflösung von 291 Inkremente / grad (Interpoliert)**

Übersetzungsverhältnis der Mechanik

Bei der Realisierung der Seilzugmechanik wurde ein maximale Übersetzungsverhältnis angestrebt, welche durch das Platzangebot im Stickgehäuse begrenzt wird.

Bei der unterschiedlichen Größe der Räder von 48 mm zu 28 mm Durchmesser, ergibt sich das Übersetzungsverhältnis zu:

48/28 = **1,714**

Skalierung in der Software

Um einen anschaulichen Messwert in der Software zu erhalten, wird der Zählerstand bei jeder Messung mit einer Konstante multipliziert, woraus sich dann der Winkel des Stick in Milligrad ergibt. Diese Konstante ergibt sich aus den Auflösung und dem Übersetzungsverhältnis zu:

K = 1/291 * 1,714 = $5{,}89 \cdot 10^{-3}$ grad/Inkremente

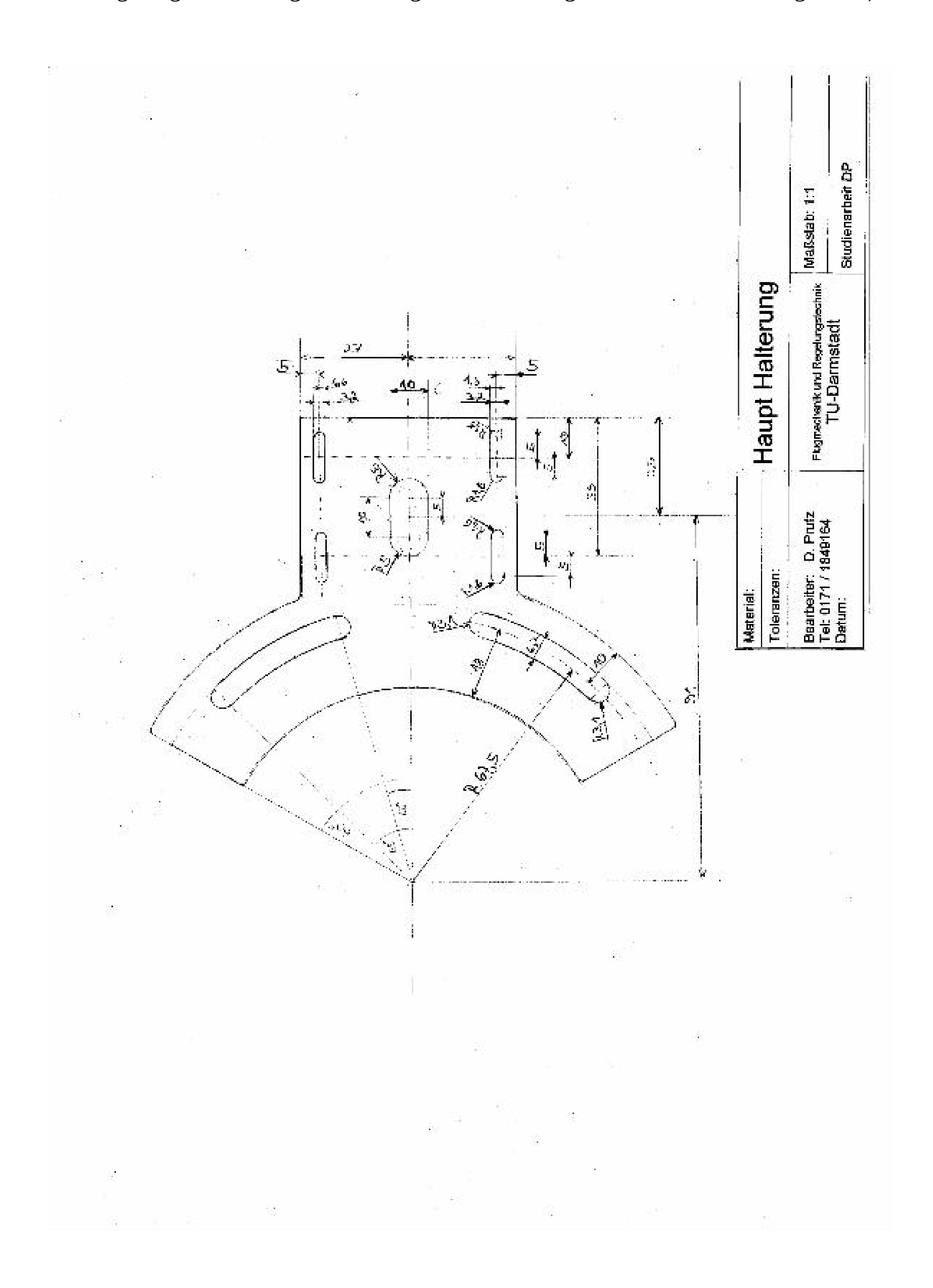
III.2 Stückliste

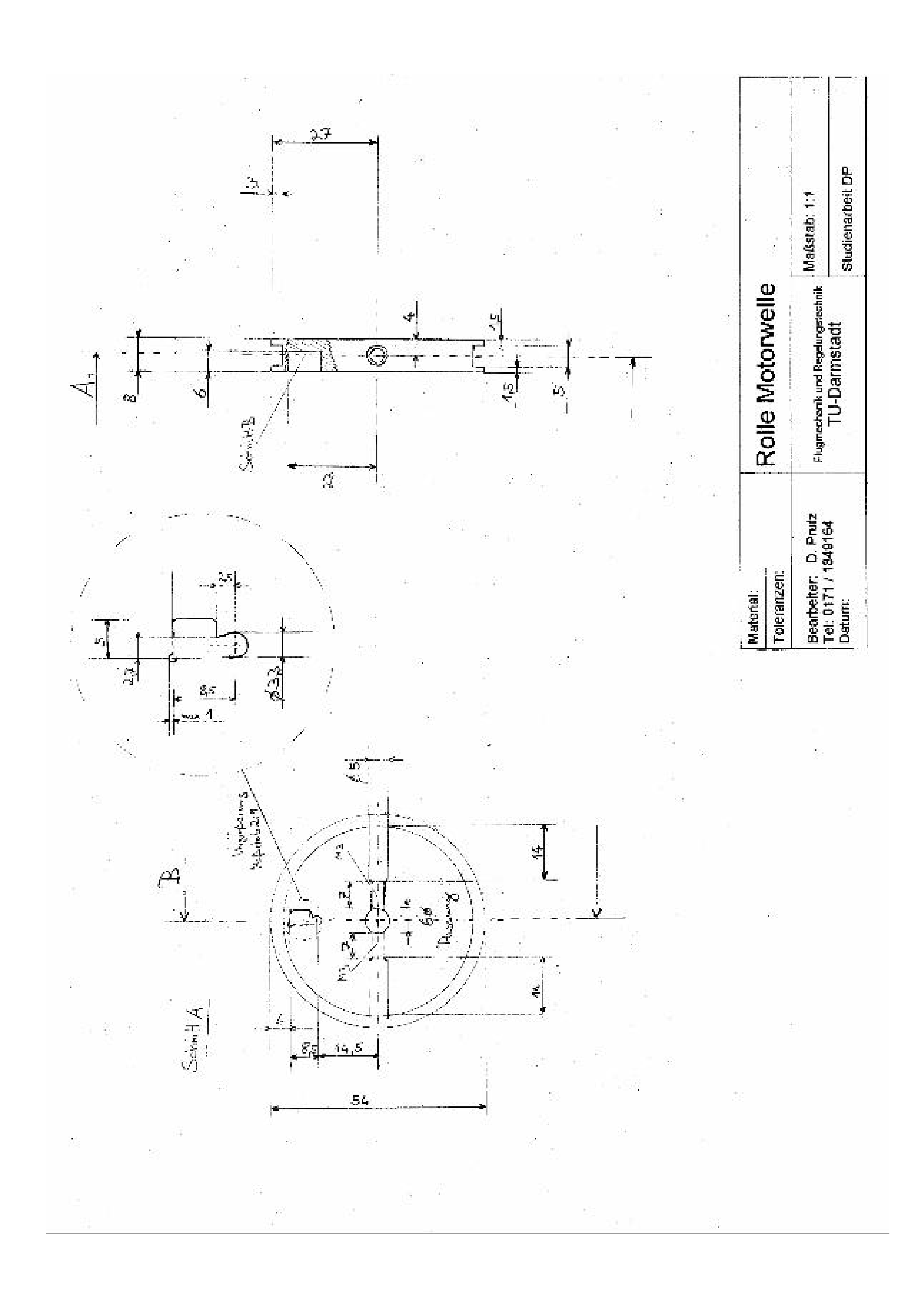
Stückliste Ink. Geber Mechanik

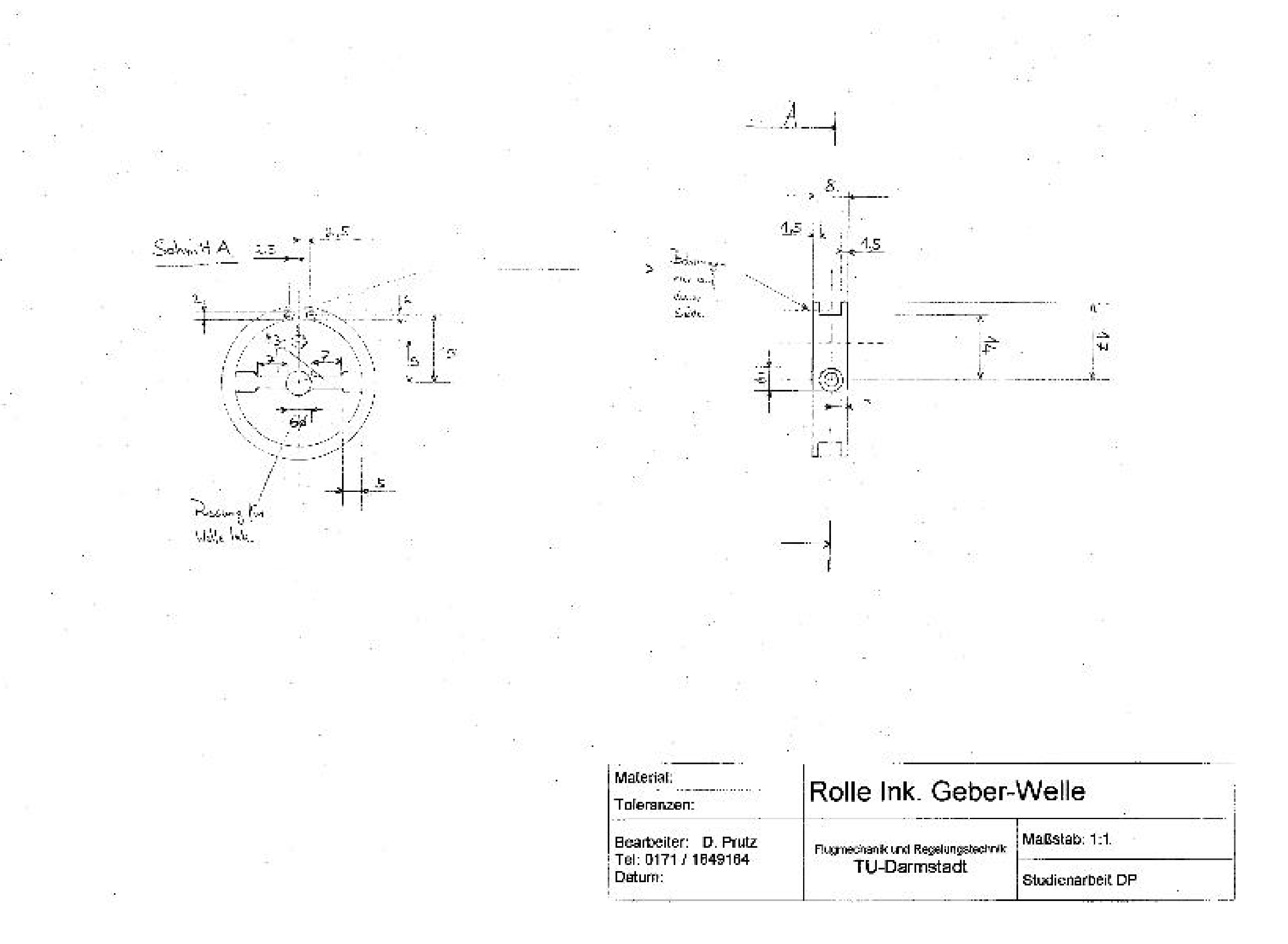
Nr.	Anzahl	Teil	
		In der Werkstadt gefertigt	
1	1	Rolle Motor	
2	1	Rolle Ink. Geber	
3	1	Haupt-Halterung	
4	1	Lagerblock Ink. Geber	
5	1	Welle Ink. Geber	
		Zukaufteile	
6	1	Wellenkupplung => Feder	
7	1m	Seil dehnungsarm	
8	2	Kugellager innen 6mm außen 19mm	
		Zubehör aus Werkstadt	
		Schrauben:	Bemerkung
9	4	M3 x 8	(für Lagerblock vorne an Haupt-Halterung (Langlöcher))
10	4	M3 x 30	(für Lagerblöcke verschraubung)
11	1	M3 x 8	(für Spannvorrichtung)
12		Madenschraube M3 x 5	
		Gewindestangen	
13	2	Gewindestangen M6 x	(als Austausch für zu kurze nach Befestigung)
		Scheiben	
14	2	Scheiben für M6	(für Langlöcher Haupt-Halterung an Motor)
15	4	Scheiben für M3	(für Langlöcher Lagerblock vorne an Haupt-Halterung
16	1	Scheibe für M3	(für Spannvorrichtung)

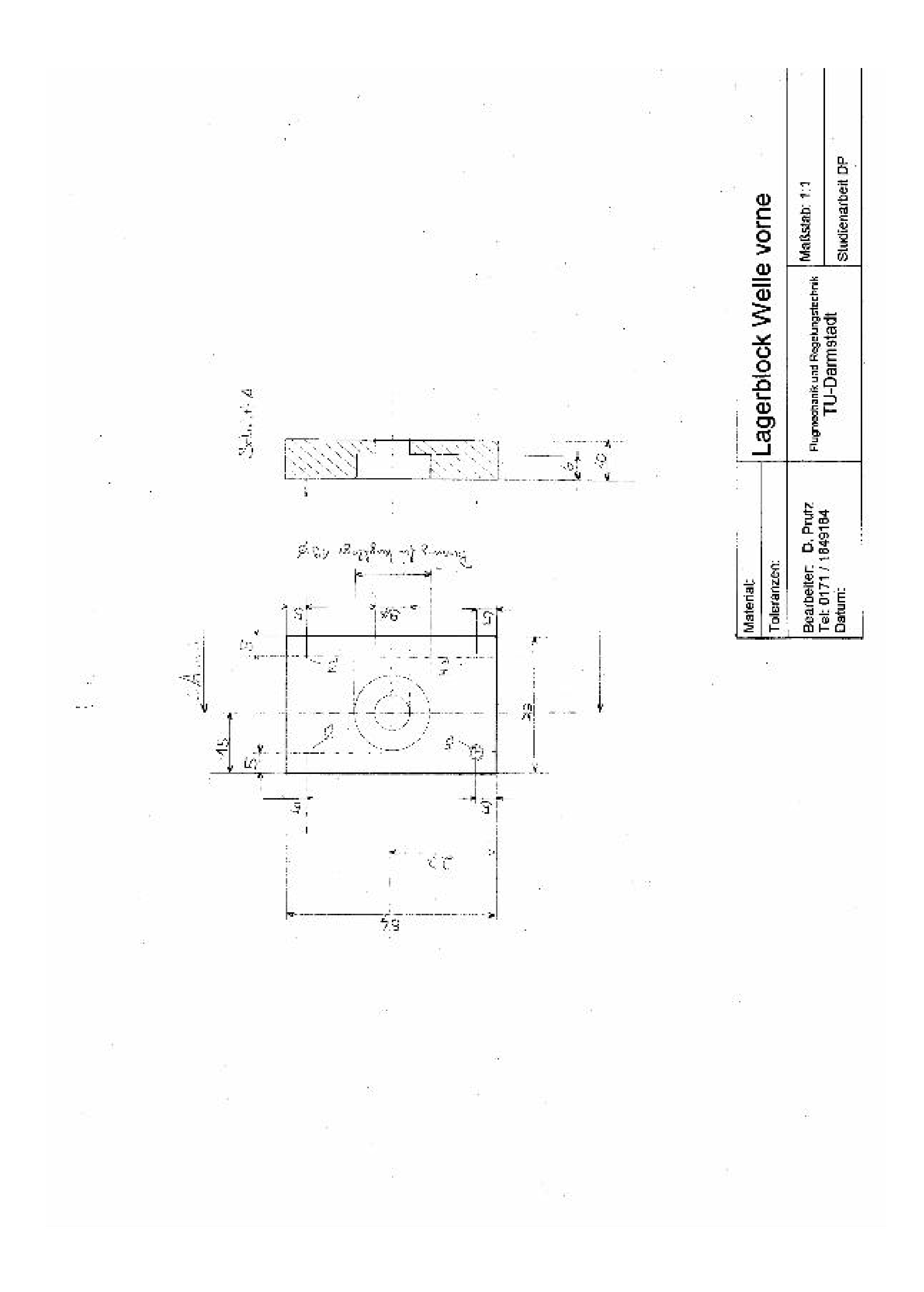
III.3 Technische Zeichnungen

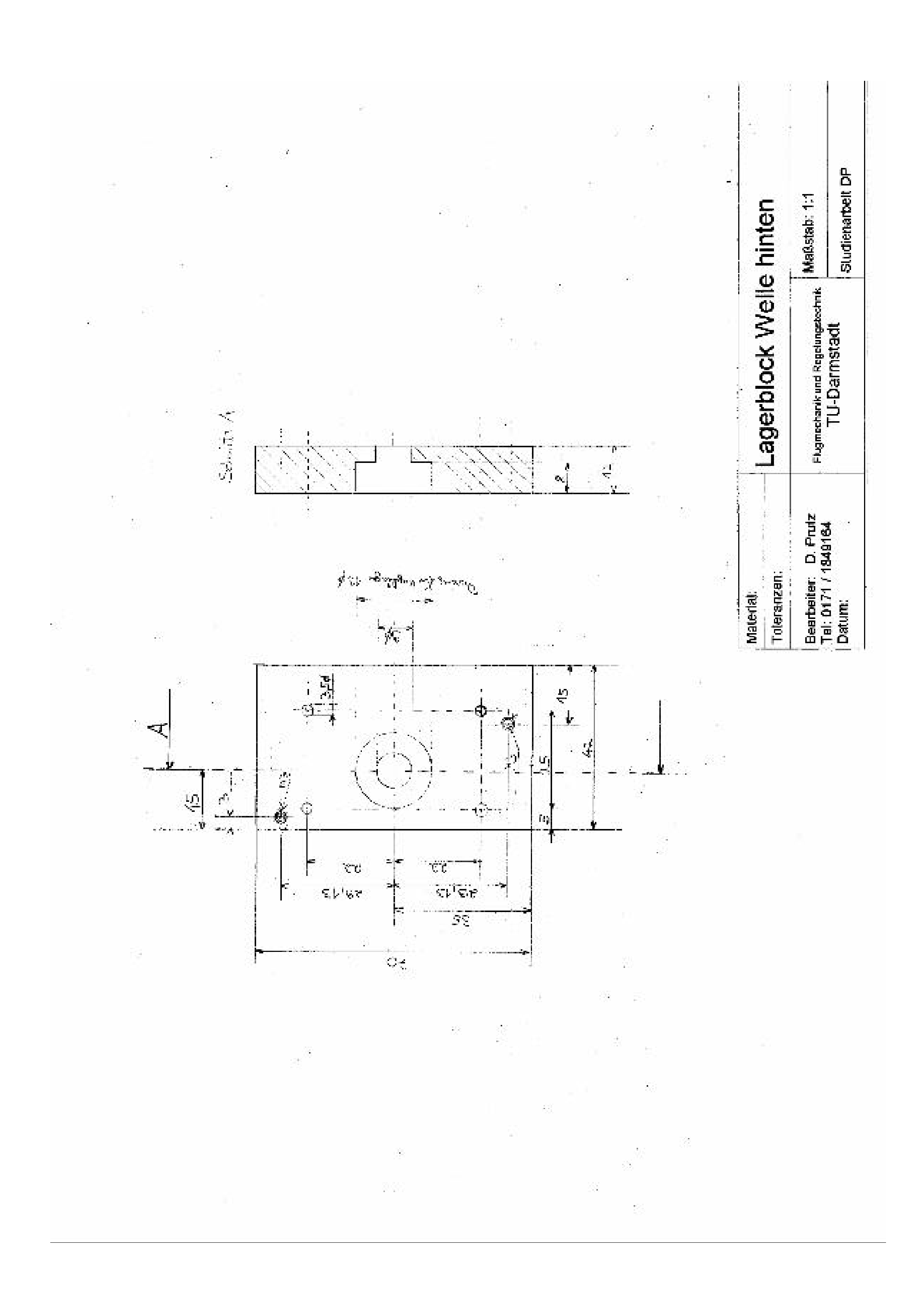
(Die Originalzeichnungen sind in der Stickdokumentation im Institut für Flugmechanik und Regelungstechnik abgeheftet. Folgende Zeichnungen sind nicht maßstabsgetreu !)

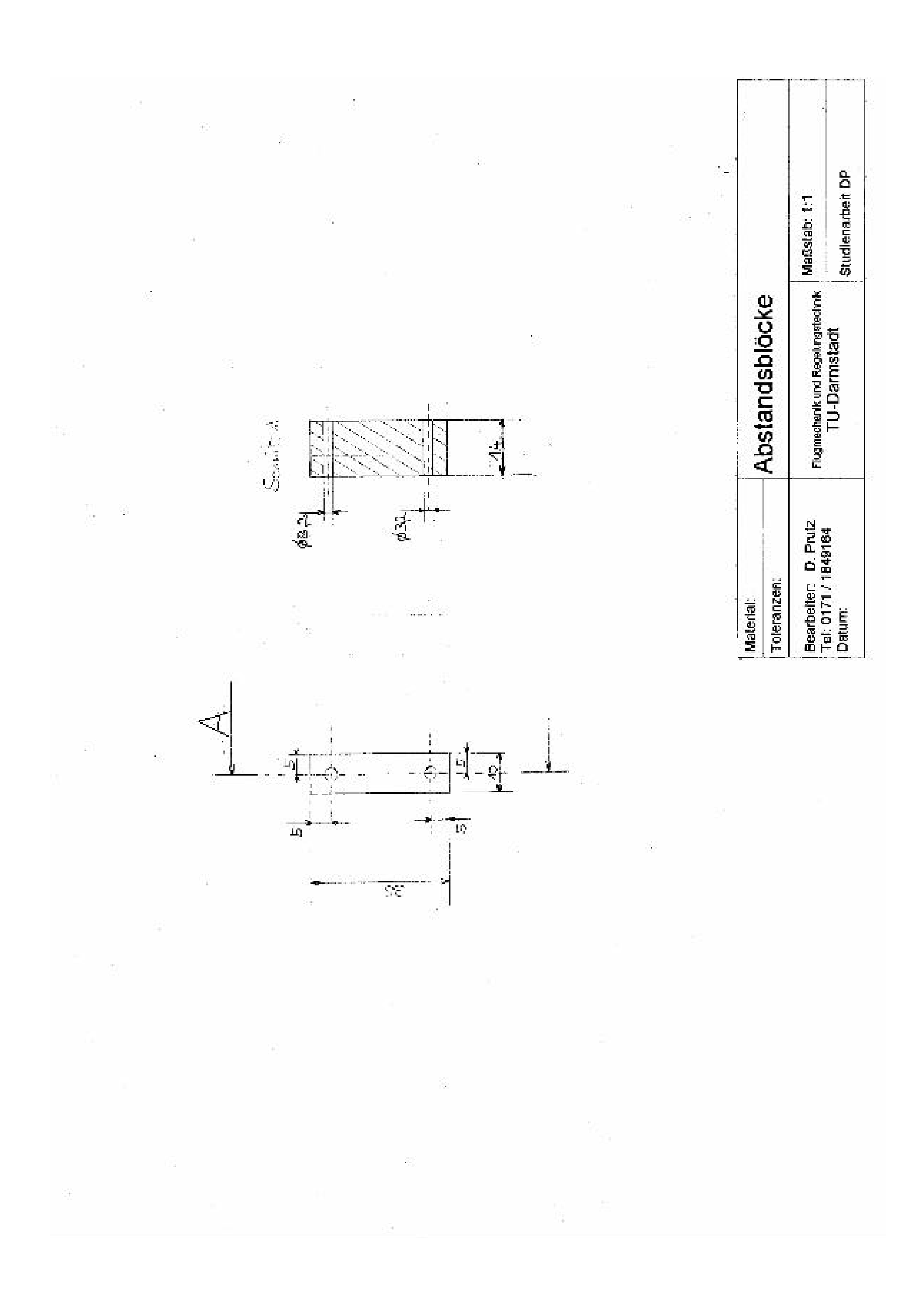


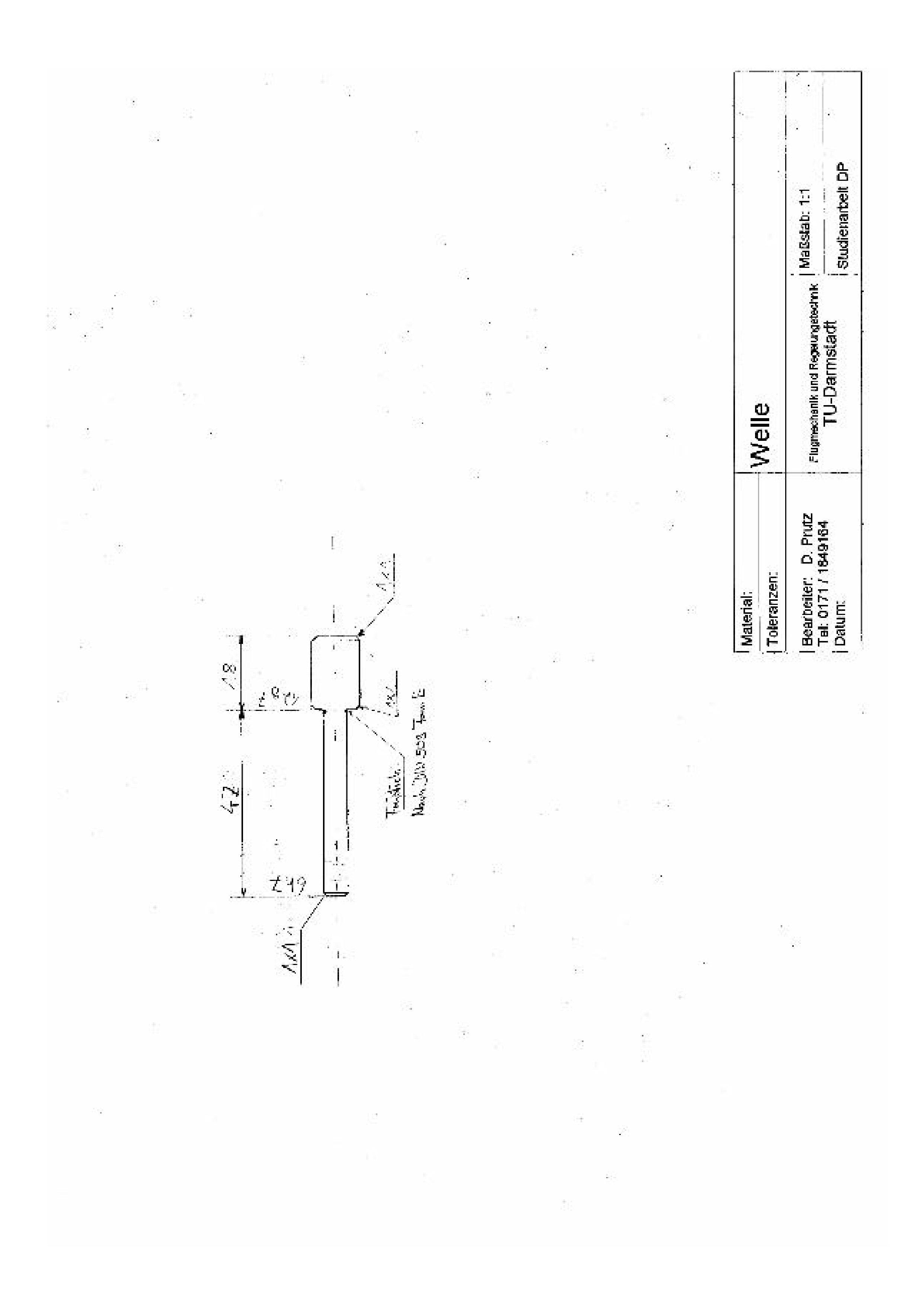

Rolle Motorwelle
Material:
Toleranzen:
Bearbeiter: D. Prutz
Tel: 0171 / 1849164
Datum:
Flugmechanik und Regelungstechnik
TU-Darmstadt
Maßstab: 1:1
Studienarbeit DP


Schnitt A
Material:
Toleranzen:
Bearbeiter: D. Prutz
Tel: 0171 / 1649164
Datum:
Rolle Ink. Geber-Welle
Flugmechanik und Regelungstechnik
TU-Darmstadt
Maßstab: 1:1
Studienarbeit DP


Lagerblock Welle vorne
Maßstab: 1:1
Studienarbeit DP
Flugmechanik und Regelungstechnik
TU-Darmstadt
Material:
Toleranzen:
Bearbeiter: D. Prutz
Tel: 0171 / 1849184
Datum:


Material:
Toleranzen:
Bearbeiter: D. Prutz
Tel: 0171 / 1849164
Datum:
Lagerblock Welle hinten
Flugmechanik und Regelungstechnik
TU-Darmstadt
Maßstab: 1:1
Studienarbeit DP
A

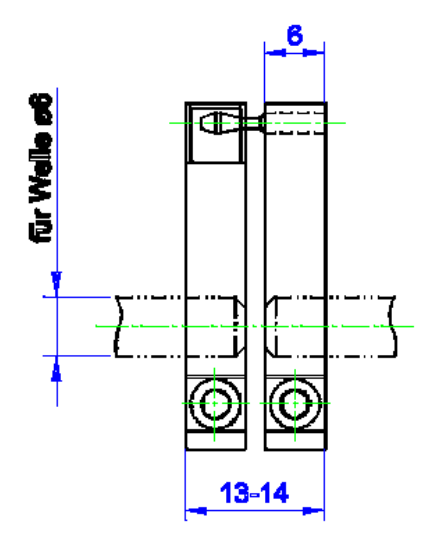

Abstandsblöcke
Material:
Toleranzen:
Bearbeiter: D. Prutz
Datum:
Flugmechanik und Regelungstechnik
TU-Darmstadt
Maßstab: 1:1
Studienarbeit DP

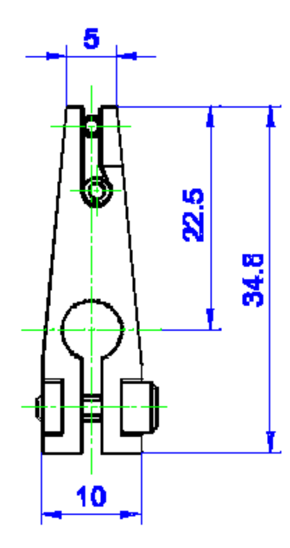

Material:
Toleranzen:
Welle
Bearbeiter: D. Prutz
Tel: 0171 / 1849184
Datum:
Flugmechanik und Regelungstechnik
TU-Darmstadt
Maßstab: 1:1
Studienarbeit DP

Für den Austausch der Kupplung und teilweisen Ersatz, wurden die technischen Daten der Potentiometerkupplung benötigt.

Hersteller: Novotechnik

Modell: z 150 g6





III.4 Umbau Stick Checkliste

Vor Vorbereitung

1) Letzte mechanische Feinheiten ausbügeln...(Senkung für Senkschrauben...)
2) Zusammenbau der Blockhalterung
   - Kugellager einpressen
   - Alles zusammenschrauben
   - Welle einpassen
   - Rolle Ink. Auf Welle montieren mit
3) Gewindestangen wenn benötigt auf richtige Länge schneiden (eventuell erst später)

Vorbereitung

4) Gehäuse Seitenplatte und Heckplatte (wenn möglich) abnehmen
5) Rechner für Datenpoolabfrage Starten
6) Stick Rechner Starten (Für Poolkommunikation)
7) Kontrolle auf einwandfreie Funktion aller Systeme

Umbau Potentiometer

8) Stick mit Hilfe von vorhandenen Befestigungen fest (in Mittelposition) fixieren
9) Im Rechenpool ablesen welcher Wert vom Potentiometer gemessen wird. => notieren
10) Stickrechner und Stromversorgung ausschalten !!
11) Potentiometer abbauen
12) Stickrechner und Stromversorgung einschalten
13) Kontrollieren, ob Potentiometer in Datenpool angezeigt wird und sich beim drehen verändert !!!
    (Grund: Es gibt für eine Achsen zwei Potentiometer !)

=>Vorsicht !!! Stick ist fixiert und es werden eventuelle Drehmomente beim verdrehen des Potis erzeugt => Daher Zentrierung des Sticks nicht aktivieren !! (Taste F11 nicht drücken))

14) Stickrechner und Stromversorgung ausschalten !!
15) Poti-Kupplung auf Motorwelle abnehmen und Feder davon in Rolle-Motor einbauen.
16) Rolle an Motorwelle aufsetzten und leicht fixieren. (mit Kupplung-Poti an relativ gleicher Position)
17) Potentiometer wieder anbringen (mit Kupplung neu an Rolle)
18) Stickrechner und Stromversorgung einschalten
19) Lage der Rolle so verändern, dass Lage des Potentiometers wieder in alter Position => mit Hilfe der Datenpool Informationen kontrollieren
20) Rolle Motor fest fixieren

Ink. Halterung anbringen

21) Ink. Halterung mit Anbau fixieren
    - dazu, wenn nötig längere Gewindestangen einbauen
22) Ink. Geber montieren => so dicht wie möglich am Motor
23) Seil auf Rollen anbringen und vorspannen (Lage der Ink. Geber Rolle dabei so, dass Seilaustritte an äußerstem Punkt des Seilzuges liegen)
24) Seil durch wegziehen der Ink. Geber Halterungen spannen

Tests

25) Stick Mechanik wieder lösen
26) Freigängigkeit des Seiles und der gesamten Mechanik überprüfen
27) Spielfreie Abnahme der Potentiometer Daten überprüfen (neue Kupplung)

IV Der Umgang mit dem Sticksystem

IV.1 Ausführbare Dateien

Vor Inbetriebnahme unbedingt lesen !!

sserve.exe und skoppel.exe

Die Datei sserve.exe und die Datei skoppel.exe sind auf den Stickrechnern in den unter IV.4 angegebenen Verzeichnissen zu finden und auch dort startbar. Beide Dateien ermöglichen die Stickregelung wie in dem Hauptteil beschrieben, wobei jedoch nur die Datei skoppel.exe auch die Drehmomentkopplung realisiert. Wichtig !!: Die Kopplung über die Stickkennlinie, wie sie früher auch mit sserve.exe möglich war, darf nicht mehr eingeschaltet werden, da diese Funktion durch den Umbau der Winkelsignale auf Drehmomentsignale nicht mehr korrekt funktioniert. Sie ermöglicht zwar bedingt eine Kopplung, jedoch treten Vorzeichenfehler und Kennlinienveränderungen auf. Daher sollte auf Experimente, vorallem auch in Kombination mit skoppel.exe verzichtet werden!

Die Dateien benötigen zum Ausführen die TRC Kalibrier Dateien Nick_trc.dat und Roll_trc.dat, sowie die Konfigurationsdatei stick.cfg (bzw. stick_2.cfg) und die stick.inp Datei für die Federkennlinie.

Beim StickPC1 muss vor dem ersten Starten der Regelung die Datei Init.exe aufgerufen werden, um die A/D Wandler zu initalisieren. Dies geschieht aber in der Regel ohnehin durch den automatischen Start von ndemo.exe nach dem Einschalten des Rechners.

ndemo.exe

Eine weitere sehr komfortable Möglichkeit die Stickregelung zu starten ist über die Batch Datei *ndemo.bat* im Verzeichnis c:\ndemo. Diese wird beim Einschalten der Stickrechner auch bereits automatisch durch einen Eintrag in der autoexec.bat aufgerufen.

Diese Datei ermöglichst eine einfache Benutzereingabe durch ein Dos Menü. Dort kann eine Kennlinie ausgewählt werden und auch die Kopplung (mit skoppel.exe) aktiviert werden.

Ndemo.exe kopiert nach Auswahl einer Kennlinie im Menue jeweils eine gespeicherte Kennlinien Datei auf die Datei stick.inp und führt danach sserve.exe oder für die Kopplung skoppel.exe aus.

IV.2 Konfigerationsdatei Stick.cfg

Folgende Tabelle soll die möglichen Eingaben in der Datei Stick.cfg erläutern, welche für die Inkrementalgeber und für die Kopplung möglich sind. Am Ende der .cfg Datei sind die Abschnitte für Inkrementalgeber (Begin Ink) und die Kopplung (Begin Koppel) unter denen folgende Variablen verändert werden können:

Tabelle IV.1. Werte für den Inkrementalgeber

Variable	Beschreibung	mögliche Werte
NickScale	Hier wird der Skalierfaktor eingetragen, der im Hauptteil aus Übersetzungsverhältnis und Auflösung berechnet wurde. Dazu kommt das Übersetzungsverhältnis der Stickmechanik. Achtung !! wenn dieser Wert verändert wird, wird auch die Dämpfung verändert !!	StickPC1 0.58877866 StickPC2 0.00058877866
RollScale		StickPC1 0.49064888 StickPC2 0.00049064888
NickDamp	Hier wird der eigentliche Dämpfungsfaktor angegeben. Tabelle IV.3 beachten !!	siehe Tabelle IV.3
RollDamp		siehe Tabelle IV.3
NickPeriode	Zur Geschwindigkeitsberechnung wird eine Feldvariable benutzt, deren Länge hier eingestellt werden kann. (Siehe 4.1)	1-30
RollPeriode		1-30
baseaddress	Basisadresse der ISA Schnittstelle, welche auf der Inkrementalgeberkarte mit Jumpern eingestellt ist.	816 (Dezimalwert) entspricht Hex 330

Tabelle IV.2 Werte für die Kopplung

Variable	Beschreibung	mögliche Werte
RollOffset	Offset Wert	je nach Anzeige auf Bildschirm
NickOffset		je nach Anzeige auf Bildschirm
RollScale	Hier wird die eigentliche Kopplung eingeschaltet: Der Wert gibt an, wie	variabel (Achtung hochfrequentes Schwingen möglich !!)

<table>
<tr><td>NickScale</td><td>groß der Einfluß der Kopplungsgröße auf die Kennlinienverschiebung des Sticks ist.</td><td>variabel (Achtung hochfrequentes Schwingen möglich !!)</td></tr>
<tr><td>RollKLsteigung</td><td rowspan="2">Gibt die Steigung der Kennlinie zur Kopplungswertübergabe an, welche unter den letzten beiden Punkten angegeben werden kann.</td><td>variabel (Achtung hochfrequentes Schwingen möglich !!)</td></tr>
<tr><td>NickKLsteigung</td><td>variabel (Achtung hochfrequentes Schwingen möglich !!)</td></tr>
<tr><td>RollKennlinie</td><td rowspan="2">Derzeit programmierte Kennlinien zur Kopplungswertübergabe:
linear
tanh (x)</td><td>1;2 (+erweiterte)</td></tr>
<tr><td>NickKennlinie</td><td>1;2 (+erweiterte)</td></tr>
</table>

Bemerkung zur Kopplung

Der Stick, an dem die Kopplung aktiviert ist, folgt jeweils dem Stick gegenüber !! (Und nicht anders herum !!)

Offset Kalibrierung

Im Input Fenster von skoppel.exe, gibt jeweils der zweite Wert von links das Drehmomentsignal des Sticks gegenüber an. Wenn die Sticks in Ruhestellung sind, kann man dort den Offset-Wert ablesen, den man in der Stick.cfg Datei als RollOffset bzw. NickOffset eingeben muss.

IV.3 Einstellbare Dämpfungswerte

Maximale Dämpfungswerte für Inkrementalgeber

Die derzeitige Problematik der hochfrequenten Schwingungen (siehe 5) führt dazu, dass bestimmte Dämpfungswerte nicht überschritten werden dürfen. Folgende Tabelle gibt diese Maximas in Abhängigkeit des softwaregenerierten Zeitintervalls der Ableitung an. Die angegebenen Minimas geben eine notwendige Dämpfung bei einer eingestellten Kennlinie von Fe 2.0 an, welche aber je nach Kennlinie variieren können.

	Nickachse Dämpfung		Rollachse Dämpfung	
T_{array}	max	min	max	Min
30	5	0,3	15	0,1
20	10	0,3	20	0,1
10	20	0,2	50	0,05
5	20	0,2	68	0,05
1	18	0,2	13	0,05

Tabelle IV.3.: Max. Dämpfungswerte für Inkrementalgeber

IV.4 Speicherort des Quellcodes auf der Festplatte

Im Laufe der Studienarbeit wurde der benötigte Quellcode der Stick-Regelung auf beiden Rechnern in einheitliche Strukturen kopiert. Der Originalcode, welcher auch noch deutlich mehr Applikationen beinhaltet wurde an seinem ursprünglichen Ort nicht verändert.

Die Speicherstruktur des aktuellen Codes, welcher sowohl die Implementierung des Inkrementalgeber als auch die mögliche Kopplung der Sticks beinhaltet wird in der folgenden Tabelle IV.4 genau erläutert.

In der Tabelle IV.5 werden die Speicherorte des Original-Quellcodes kurz dargestellt.

Verzeichnis	Beschreibung	Bemerkung
C:\stick1\ (auf StickPC1) C:\stick2\ (auf StickPC2)	Hauptverzeichnis des Quellcodes für Stick-Regelung	Aktuell , inklusive der Ergänzungen der Studienarbeit
..\lib	Kompilierte Libary-Files der Module (.lib)	
..\include	Include Files der Stick-Regelung	
..\source	Hauptverzeichnis des Quellcodes der Module	
..\source\driver	Module für die A/D Wandler Karte (Nicht auf StickPC1 !! Da andere Hardware !)	Kompilierung ergibt c:\usr\stick2\lib driver_l.lib
..\source\low_lvl	Module für die Initialisierung und Parameter einlesen	Kompilierung ergibt c:\usr\stick2\lib ad_dma_l.lib
..\source\kennl	Module für die Stick Regelung	Kompilierung ergibt c:\usr\stick2\lib stick_l.lib
..\sourcek	Hauptverzeichnis des Quellcodes der Module für die Kopplung	
..\sourcek\low_lvl	Module für die Initialisierung und Parameter einlesen (Mit Kopplung)	Kompilierung ergibt c:\usr\stick2\lib ad_dma_l.lib

..\sourcek\kennl	Module für die Stick Regelung (Mit Kopplung)	Kompilierung ergibt c:\usr\stick2\lib stick_l.lib
..\tools	Benötigte Dateien für Applikationen	(nur .lib und include files)
..\applic\sserve	Code für Hauptanwendung sserve.exe (kann in diesem Verzeichnis auch ausgeführt werden, da alle weiteren erforderlichen Konfigurations-Dateien dort vorhanden sind)	Sserve.exe beinhalten die gesamte Stick-Regelung mit Ink-Geber-Dämpfung
..\applic\skoppel	Code für Hauptanwendung skoppel.exe (kann in diesem Verzeichnis auch ausgeführt werden, da alle weiteren erforderlichen Konfigurations-Dateien dort vorhanden sind)	Skoppel.exe beinhalten die gesamte Stick-Regelung mit Ink-Geber-Dämpfung und der Kopplung beider Sticks

Tabelle IV.4 : Aktuelle Speicherorte des Quellcodes für die Stickregelung

Auf Stick PC 1		
C:\stick	Hauptverzeichnis des Quellcodes für Stick-Regelung (Die weitere Aufteilung der Verzeichnisse entspricht größtenteils dem Verzeichnis c:\stick1) Es sind auch noch andere Applikationen und Module dort abgespeichert.	Aktuell , jedoch ohne Studienarbeit Ergänzungen
C:\usr\Tommy\	Weitere und doppelte Quellcodes zur Stickregelung	Stellenweise Nicht aktuell !!
C:\usr\Frank\	Code für Applikation sserve (Stand vor der Studienarbeit)	
Auf Stick PC 2		
C:\usr\stick1	Hauptverzeichnis des gesamten Quellcodes für StickPC1 (nur ältere Kopien)	Nicht aktuell !! (nur ältere Kopien)

C:\usr\stick2	Hauptverzeichnis des gesamten Quellcodes für StickPC2 (Die weitere Aufteilung der Verzeichnisse entspricht dem Verzeichnis c:\stick2) Es sind auch noch andere Applikationen und Module dort abgespeichert.	Aktuell, ohne Studienarbeit Ergänzungen

Tabelle IV.5 : Aktuelle Speicherorte des Original-Quellcodes für die Stickregelung

IV.5 Wichtige Programmteile der Sticksoftware

Einlesen der Datei Stick.cfg (Stick_2.cfg)			
Get_sys print_channels ()	C Void	\Source\Low_Lvl	Initalisierung der Konstanten (umschreiben aus Datei in Global definierte Variablen)
Messwert Abruf der Hardware			
Komukomp	C	\Source\Kennl	Hardware Messwertaufnahme der IO Karten
Get_data	Func		Abruf aller Meßwerte der IO Karten (inklusive der Ink. Gebers)
Ablaufsteuerung			
Stick.C	C	\Source\Kennl	Eigentliche Steuerdatei mit Summationspunkt des Regelkreislaufes
Stickinit	Func		Initialisieren der IO Karten
RollStick	Func		Regelmodul für Rollachse
NickStick	Func		Regelmodul für Nickachse
Sserve.C	C	\applic\sserve	Einbindung aller Module und Bildschirmausgabe
ReglerRoll	Func		Regler Rollachse (derzeit kein Regler benutzt)
ReglerNick	Func		Regler Nickachse (derzeit kein Regler benutzt)
OutpWatch	Void		Ausgabefenster der Ausgänge
InpWatch	Void		Ausgabefenster der Eingänge
Kennlinien Auswertung und Verarbeitung			
Name	Typ	Ort	Beschreibung
K.L	Lex	\Source\Kennl	Kennlinien .inp File Interpretation
parser	Func		Stick Kennlinien Auswertung => Steuerung z.B. auch von wo die Kennlinie eingegeben wird.
KLinie ( )	Func		Kennlinien Auswertung und Umrechnung in Größe für die Regelfunktionen

IV.6 Programmteile der Inkrementalgeber

Datei	Funktionen /Proceduren	F/P	Übergabewerte/Rückgabewert
IK121out.c	init_ik121	P	(baseaddress)
	getvalue_ik121	F	(int base_address, int axes) (double)
	arrayspeed_ik121	F	(int axes) (double)
	Inkspeed	F	(int axes) (double)
	Timetest	F	(int index) (double)
IK121_0.c	Beschrieben in Heidenhain Anleitung [X]	P/F	

IV.7 Kompilieren des Codes in .lib / .exe

Bei dem verwendeten Borland C 3.0 (StickPC1) und 4.0 (StickPC2) Compiler ist die Kompilierung nur durch das manuelle Erstellen eines „makefiles“ (in den neueren grafikorientierten Tools auch Projektfile genannt) möglich.

Dieser makefile beihalten sämtliche Informationen über Speicherort, Quelldateien, Include Dateien und Compiler Erweiterungen, die für das Erstellen einer Libary (.lib) oder einer ausführbaren Datei (.exe) nötig sind.

Die genaue Syntax ist compilerabhängig und wird in der jeweiligen Compilerdokumentation erläutert.

Für die Einbindung eines neuen Quellcodes zum Kompilieren (wie es für die Inkrementalgeber und Kopplung Implementierung notwenig war), muss lediglich der Dateinahme des Sourcefiles in der „SOURCE“ und „OBJS“ Option mit der jeweiligen passenden Endung aufgelistet werden und der dazugehörige Include File in der Option „HEADERS“ erwähnt werden.

IV.8 Unterschiede der Sticksysteme

Wie bereits im Hauptteil stellenweise verdeutlicht wurde, sind die Sticksysteme des Piloten und des Copiloten zwar vom Konzept her gleich, unterscheiden sich jedoch in bestimmten Teilen der Hardware und dadurch resultierend auch in der Software. Diese Unterschiede soll die folgende Tabelle erläutern:

Eigenschaft	bei beiden gleich	unterschiedlich
Aufbau der Aktoren, Sensoren und Verstärker	Typ und Anordnung der Motoren und Winkelsensoren (Potentiometer)	Aufbau und Ort der Messverstärker bei Stick1 ist alles im Stickgehäuse untergebracht. bei Stick2 ist alles an einem extra Kasten am Stickgehäuse montiert. Die Drehmomentsensoren + Verstärker sind unterschiedliche Typen
Mechanik	Übersetzungsverhältnis der Achsen	Lagerung Stick1 = Kugellager Stick2 = Wälzlager (viel reibungsfreier !!
A/D Wandler	beide 16 Bit Abtastung mit 1kHz (durch Software vorgegeben)	Stick 1: Im Stickschrank untergebracht und von deutlich älterer Bauart. Zusätzlich I/O Karte in Stick PC Stick 2: Im Stick PC als ISA Schnittstellenkarte
StickPC	Betriebssystem RT-Dos	Stick1: 80486 120MHz Stick2: Pentium 90
Kompilersoftware	Borland C	Stick1: Version 3.0 Stick2: Version 4.0
Sticksoftware	Regelalgorithmus Treiber für Inkrementalgeberkarte	Hardware Treiber für A/D Karten

Tabelle IV.6 : Unterschiede der Sticksysteme

V Kurzbeschreibung der Erweiterungen

Zusätzliche Variablen durch die Inkrementalgeber Software Einbindung

Globale Variablendefinition in Datei [Ik121out.h]

```
typedef struct {
        double wert;
        double scale;
        double damp;
        double speed;
        unsigned int periode;
        } Ink;

extern Ink InkRoll, InkNick;
extern Int baseaddress;
```

Globale Variablendefinition in Datei [Ik121out.c]

```
Ink InkRoll, InkNick
Int baseaddress;
```

Statische Variablendefinition in Datei [Ik121out.c]

```
static double Ink_oldvalue_roll;
static double Ink_oldvalue_nick;
static double Ink_array_rollvalue[31];   /* Ink Achse 1 = roll Achse */
static double Ink_array_nickvalue[31];   /* Ink Achse 2 = nick Achse */
static unsigned int index1=1;
static unsigned int index2=1;
```

Erweiterung der Sticksoftware um die Inkrementalgeberfunktionen

Originaldatei	Eingefügt	Zweck
Einfügen des extra Quellcodes für Inkrementalgeber		
IK121out.c	\source\low_lvl	Zusätzliche Funktionen für Sticksoftware
IK121_0.c	\source\low_lvl	Grundfunktionen für Ink. Geber
IK121out.h / IK121_0.h	\include	Funktionendeklaration (+Variablendeklaration)
Definition der Konstanten aus der Datei Stick.cfg		
Stick.cfg	Zusatz Skript	Definition von Konstanten für Ink. Geber
get_sys.c	Zusatz Quellcode	#Include <ik121out.h>

get_sys.c	Zusatz Quellcode	Lesen von Daten aus Stick.cfg
Initalisierung der Inkrementalgeberkarte		
Stick.C	Zusatz Quellcode	#Include <ik121out.h>
Stick.C => StickInit	Init_ik121(..)	Initialisierung der Register, etc.
Abrufen der Ink. Geber Messdaten in der Stick Software		
komukomp.c	Zusatz Quellcode	#Include <ik121out.h>
komukomp.c	Zusatz Quellcode	Globale Variablendeklaration für Funktion arrayspeed_ik121
komukomp.c => get_data	Getvalue_ik121	Abruf Ink. Meßwert
komukomp.c => get_data	arrayspeed_ik121	Berechnung der Geschwindigkeit
Bildschirmausgabe		
Sserve.c	Zusatz Quellcode	#Include <ik121out.h>
Sserve.c => OutpWatch	Zusatz Quellcode	Wprintf (...InkRoll.wert, InkNick.wert)

Zusätzliche Variablen durch die Kopplung

Globale Variablendefinition in Datei [koppel.h]

```
typedef struct Koppel
        {
        double wert;
        double offset;
        double scale;
        double steigung;
        int kennlinie;
        } Koppel;

extern Koppel Rollkoppel,Nickkoppel;
```

Globale Variablendefinition in Datei [koppel.c]

```
Koppel Rollkoppel,Nickkoppel;
```

Erweiterung der Sticksoftware zur Realisierung der Kopplung

Originaldatei	Eingefügt	Zweck
Einfügen des extra Quellcodes für Inkrementalgeber		
Koppel.c	\source\kennl	Zusätzliche Funktionen zum Berechnen der Kopplungwerte
Koppel.h	\include	Funktionendeklaration +Variablendeklaration
Definition der Konstanten aus der Datei Stick.cfg		
Stick.cfg	Zusatz Skript	Definition von Konstanten für Ink. Geber
get_sys.c	Zusatz Quellcode	#Include <koppel.h>
get_sys.c	Zusatz Quellcode	Lesen von Daten aus Stick.cfg
Implementieren des Koppelsignals		
Stick.C	Zusatz Quellcode	#Include <ik121out.h>
Stick.C => StickInit	Koppelnick() Koppelroll()	Zusätzliche Summation der Kopplung in StickRegelung
Bildschirmausgabe		
Sserve.c	Zusatz Quellcode	#Include <koppel.c>
Sserve.c => InpWatch	Zusatz Quellcode	Wprintf (...RollData.passiv,...) Wprintf (...NickData.passiv,...)

VI Inkrementalgeber Programmierung

VI.1 Testsoftware Beschreibung

Für die Implementierung der Ink. Geber als Geschwindigkeitsmesser, musste als erstes einige Messungen vorgenommen werden, um ein erste Aussagen über den Zeitbereich treffen zu können, in dem später differenziert werden sollte. Dazu wurden erste Programme (für die Tests noch in Pascal) geschrieben, mit dem Ziel eine Messreihe auf zu nehmen, in der mit unterschiedlichen Zeiten differenziert wird. Diese Messreihen sollten dann möglichst anschaulich in einem Graphen sichtbar gemacht werden. Da es sich bei den aufgenommen Messungen um teilweise bis zu 4*32000 Werte handelte, musste ein Programm geschrieben werden, was die spätere Verarbeitung möglichst automatisiert. Dies wurde realisiert, indem die Importfunktion von MS Excel benutzt wurde, um eine strukturierte Datei einzulesen. Daher mussten die Werte der Messungen nur strukturiert in eine Text Datei geschrieben werden, so das Excel diese ohne große Probleme importieren konnte.

In dieser ersten Programmierphase, die sich ausschließlich mit dem Verstehen und darauf aufbauend dem Programmierung der Ink. Geber Karte befasste, wurden auch die Algorithmen zur Ink. Geber Karten Steuerung erkannt und beschrieben. Flussdiagramme siehe dazu VI.2.

Erste Testprozeduren:

In ersten Testprozeduren (z.B. DPTest1) sind beschriebene Abläufe genau nachzuvollziehen, wobei dort jeder Schritt bereits für jede einzelne Achse getätigt wird.

Weitere Testprozeduren wie „DPSpeed2.pas“ geben weiter Beispiele für die zusätzliche Verarbeitung der Messwerte und auch schon das Abspeichern in eine Datei.

Prozeduren für die Tests:

Die wichtigsten Prozeduren, die auch während der Testaufnahme benutzt wurden sind die Proceduren „Stickweg.pas“, „StickGes.pas“ und „Stickall.pas“. Der Unterschied zu den Testprozeduren besteht darin, dass diese Prozeduren alle Daten als Record in eine einzige Datei abspeichern und diese Datei auch beim Beenden des Tests zum Lesen in Excel konvertiert wird.

Konvertieren der Daten für Excel:

Um eine sinnvolle Auswertung der aufgenommenen Daten zu erreichen, sollten die in Dateien abgespeicherten Daten relativ leicht über die Importfunktion von Excel eingelesen werden. Dazu wurden die Pascal Units „Convert.pas“ bzw. „Converta.pas“ mit der Prozedur convert_dat_txt geschrieben.

Das erste Hauptproblem besteht allerdings darin, dass sämtliche Daten, die in einem Pascal oder C Programm in eine Datei abgespeichert werden, immer als „Charakter“ interpretiert werden, und damit automatisch in ASCII Darstellung abgespeichert werden. Dies heißt, dass die Datei nur zum Wiedereinlesen in Pascal bzw. C benutzt werden kann, allerdings nicht zum direkten Ablesen der Daten bzw. Messwerte. Daher musste eine Prozedur geschrieben werden, die die gewonnenen Daten in eine neue Datei konvertiert, in der die Messreihen mit einem Texteditor oder eben Excel direkt ablesbar sind. Ist man sich der Tatsache bewusst, dass innerhalb eines Programms alles in ASCII Format abgespeichert wird, so ist die Konvertierung zwar ein wenig umständlich aber keineswegs problematisch. Es werden dazu lediglich die Daten aus der während der Messung gewonnenen Datei wieder eingelesen und jeder einzelne dezimale Messwert erst in seine einzelnen Stellen zerlegt, um diese dann einzelnd mit dem entsprechenden ASCII Format in die neue Datei zu schreiben. So muss z.B. für eine richtige „1“ in der Datei eine „49“ in die Datei geschrieben werden, für eine „2“ eine „50“ und so weiter.....weiterhin wird nach dem Durchgang aller Messwerte des gleichen Zeitpunktes (z.B. Winkel Achse1, Winkel Achse2, Speed Achse1, Speed Achse2) eine ASCII Zahl für „Return“ eingefügt, um die Daten nicht endlos aneinander zu schreiben, sondern in einer fortlaufenden Spalte zu führen.

Ist die Konvertierung, die aus Laufzeitgründen erst nach der Messwertaufnahme erfolgt, abgeschlossen, kann die neu gewonnen .txt Datei mit einem Mausklick in Excel importiert werden, und liefert so mit einem oder auch zwei weiteren Mausklicken einen sehr repräsentative Darstellung in Graphenform. => siehe dazu Auszüge im Anhang II bzw. vollständige Ergebnisse in der Datei „Alle Messwerte.xls“

Übersicht der Funktionen und benötigten Units:

Aufgrund der unterschiedlichen Anzahl von Messwerten, müssen für die einzelnen Programme auch unterschiedliche Units verwendet werden, worüber folgende Tabelle eine Übersicht geben soll:

Programm	selbst erstellte Units	sonstige benötigte Units	Anzahl der Messwerte pro Messung
Stickweg.pas	Convert.pas	Crt.pas / ik121.pas	2
Stickges.pas	Convert.pas	Crt.pas / ik121.pas	2
Stickall.pas	Converta.pas	Crt.pas / ik121.pas	4

Berechnungen zu Testaufnahmen

Innerhalb der Testsoftware wird die Ableitungszeit über den Befehl „delay" realisiert. Da jedoch der Delay-Befehl in Pascal prozessorabhängig ist, wurde eine Messreihe durchgeführt, um die Delay-Werte beim benutzten P75 PC in ms umrechnen zu können. Die Ergebnis sind in der folgenden Tabelle dargestellt. Weiterhin wurde bei den gegebenen Delay-Zeiten die Dateigröße pro Sekunde zu ermittelt, welche am Ende des Testlaufes auf eine Diskette passen sollte.

Aufzeichnungen der Ink Geberwerte nur Weg

Delay Time	Werte pro Sekunde (2*Weg)	Größe der Datei pro Sekunden (kByte)	Maximale Aufzeichungsdauer bei Größe 1,3MB	
			Sekunden	Minuten
0	4500	70,4	18	0,31
1	2700	43,1	30	0,50
10	730	11,5	113	1,88
90	100	1,57	830	13,83
100	87	1,4	929	15,48
600	14	0,23	5571	92,86
900	10	0,16	8298	138,30

Aufzeichnungen der Ink. Geberwerte Weg und Geschwindigkeit

Delay Time	Werte pro Sekunde 2*X+2*V	Größe der Datei bei 10 Sekunden (kByte)	Maximale Aufzeichungsdauer bei Größe 1,3MB	
			Sekunden	Minuten
0	4500	140,8	9	0,15
1	2700	86,2	15	0,25
10	650	20,4	64	1,06
90	100	3	433	7,22
100	83	2,7	481	8,02
600	14	0,47	2786	46,43
900	10	0,31	4149	69,15

Delay Wert für P75

Zeit Delay werte:	Reale Zeit	Frequenz
	ms	Hz
0,9	0,1	10000
9	1	1000
90	10	100
900	100	10
600	1500	0,67
9000	1000	1

Tests:

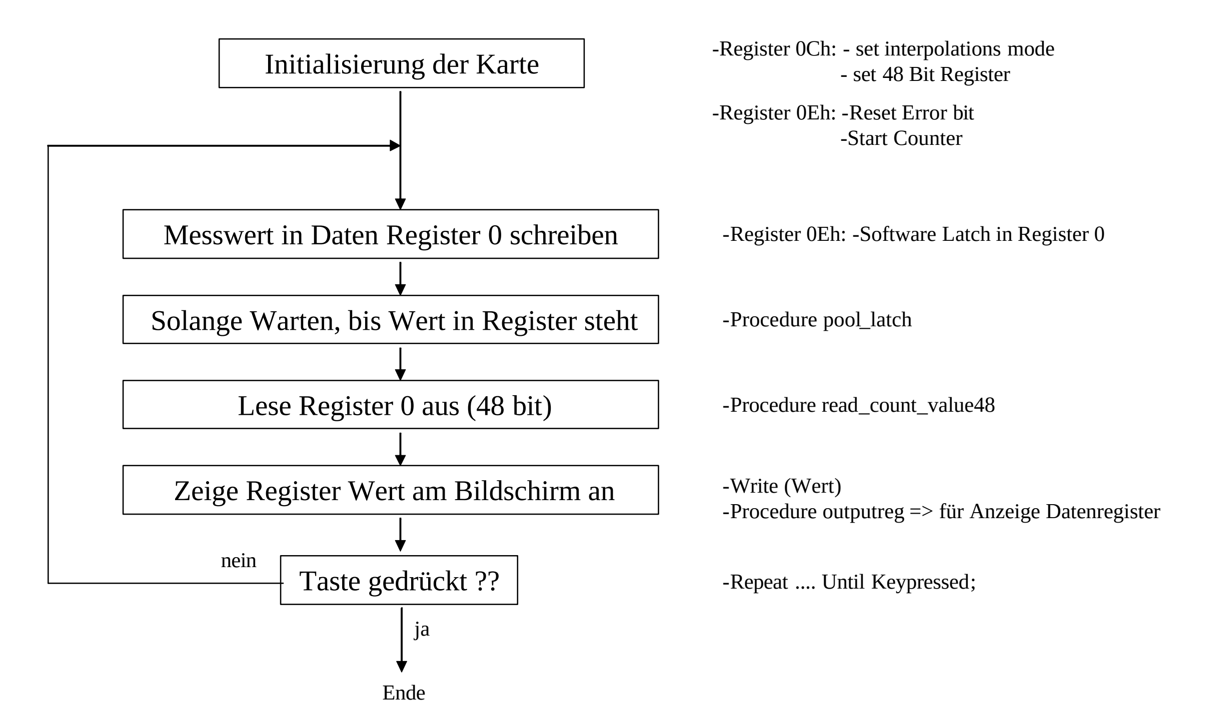
=> zur Ermittlung einer sinnvollen Datenrate
1) Langzeittest von nur Position => time = 90
2) Kurzzeittest von nur Position => time = 0

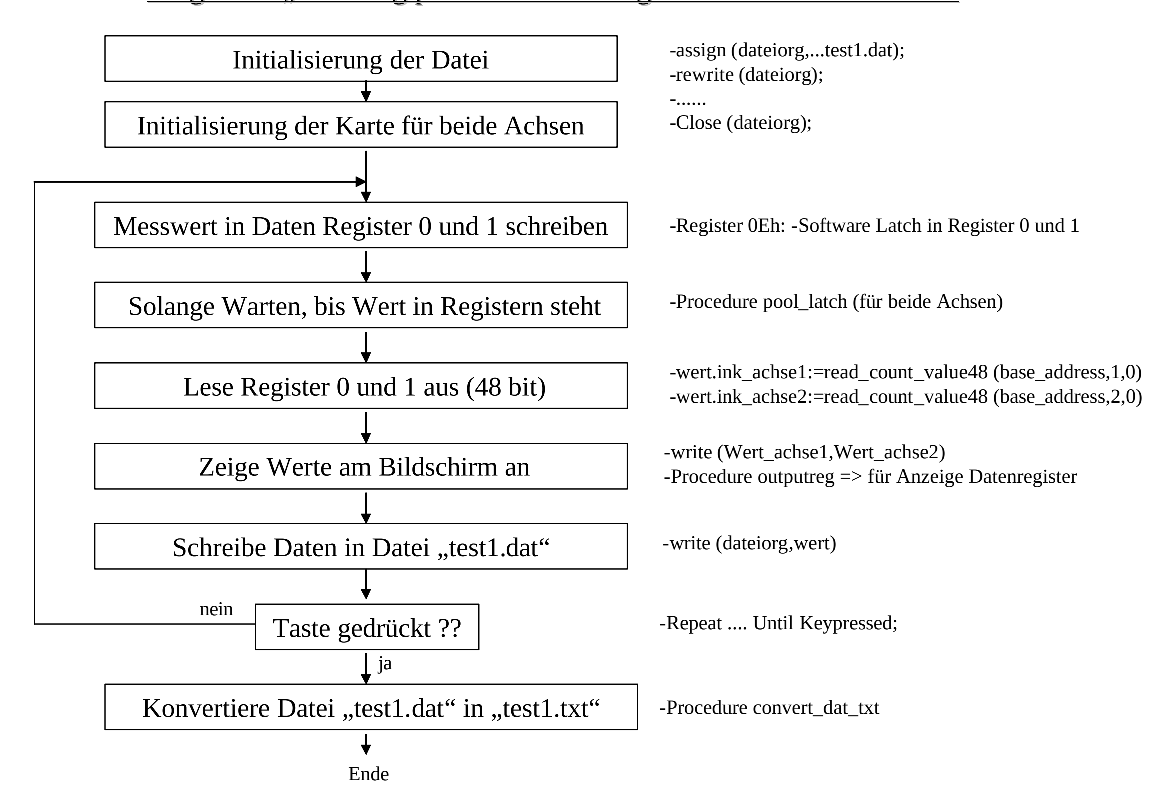
=> zur Ermittlung einer Sinnvollen time für beste Geschwindigkeitsaufnahme
3) Test mit unterschiedlichen Zeiten
=> time = 30
=> time = 60
=> time = 90
=> time = 180
=> time = 300
=> time = 600
=> time = 900

Prozedur zur Ausgabe der Register:

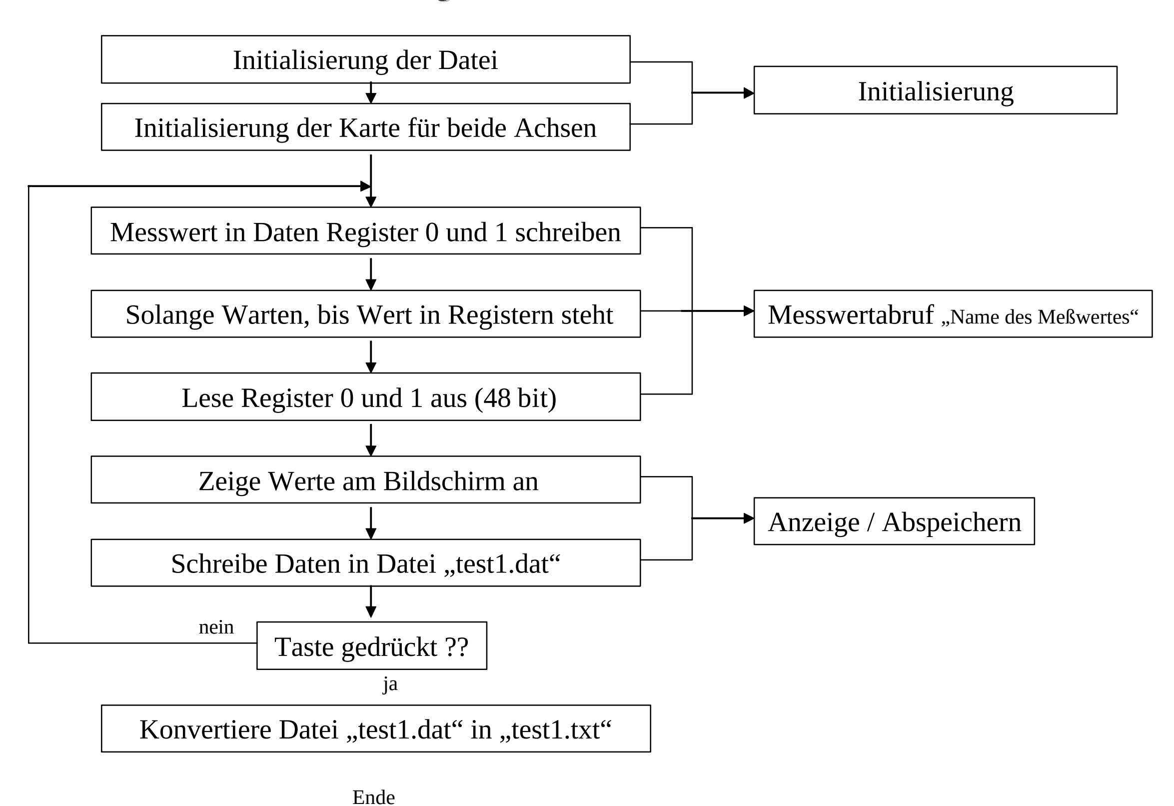
Um einen ersten Eindruck und auch Überprüfung der Register innerhalb des laufenden Betriebs zu bekommen, wurde die Pascal Prozedur „outputreg" geschrieben, welche in der Unit „Control" vorhanden ist. Diese Prozedur ohne Übergabevariablen kann einfach aufgerufen werden und gibt dann alle lesbaren Register der Karte auf den Bildschirm aus. Um eine Darstellung der einzelnen Bits zu erreichen wurde innerhalb der Unit „Control" noch eine Prozedur geschrieben, welche die hexadezimale Darstellung der Register in binäre Darstellung umwandelt, und dadurch erst die einzelnen Bits der Register abgelesen werden können.

VI.2 Flussdiagramme zur Testsoftware

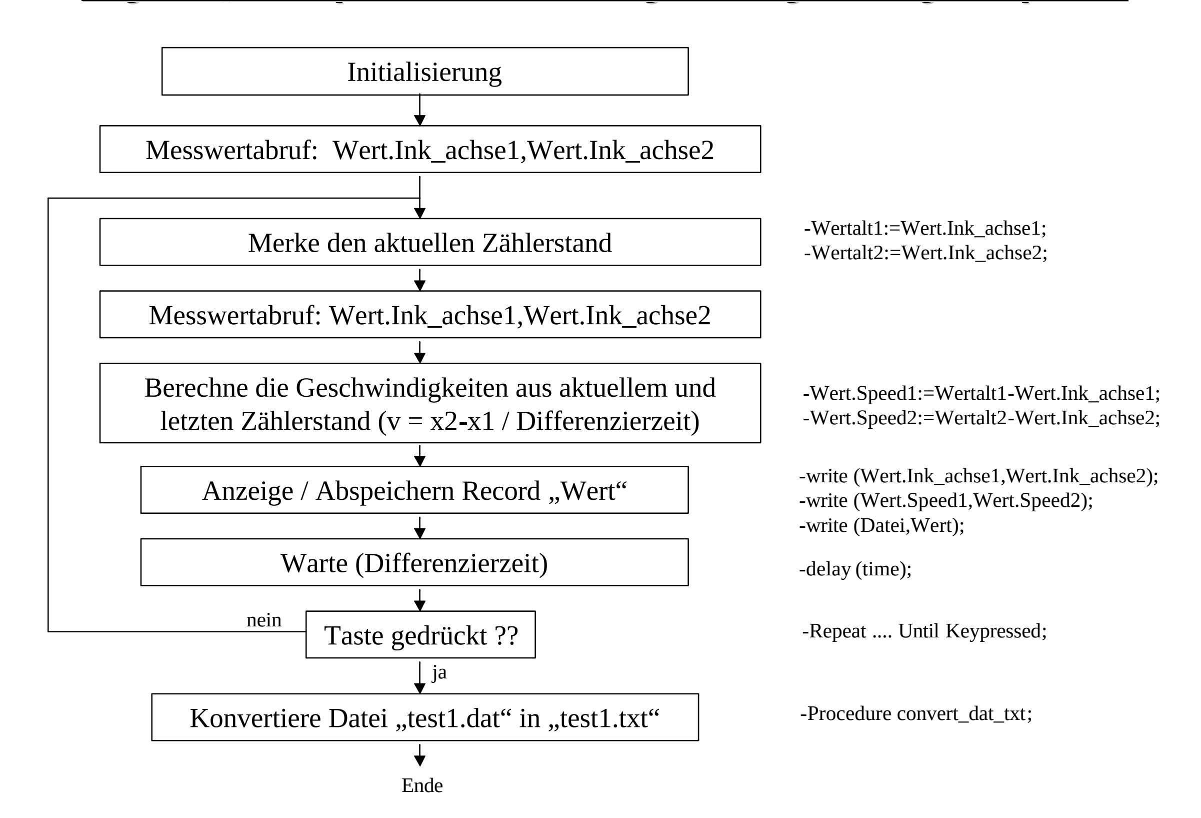




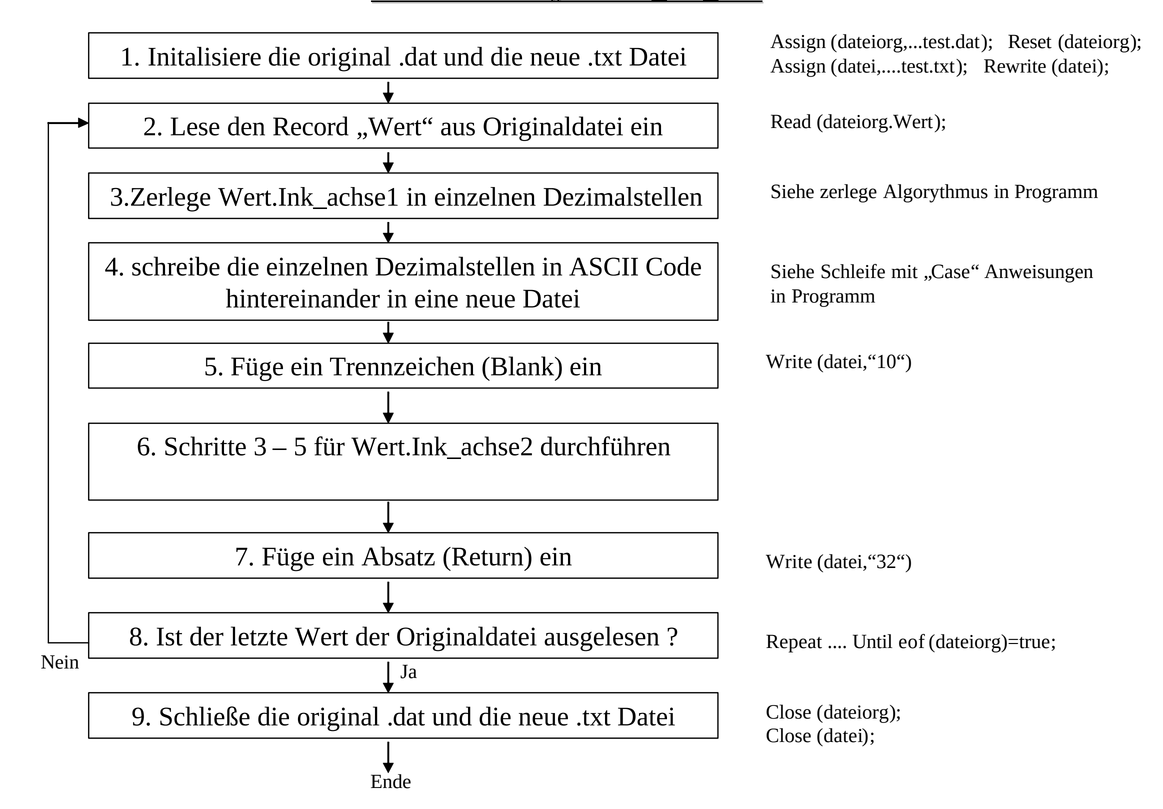
Zusammenfassung der Funktionen für mehr Übersichtlichkeit



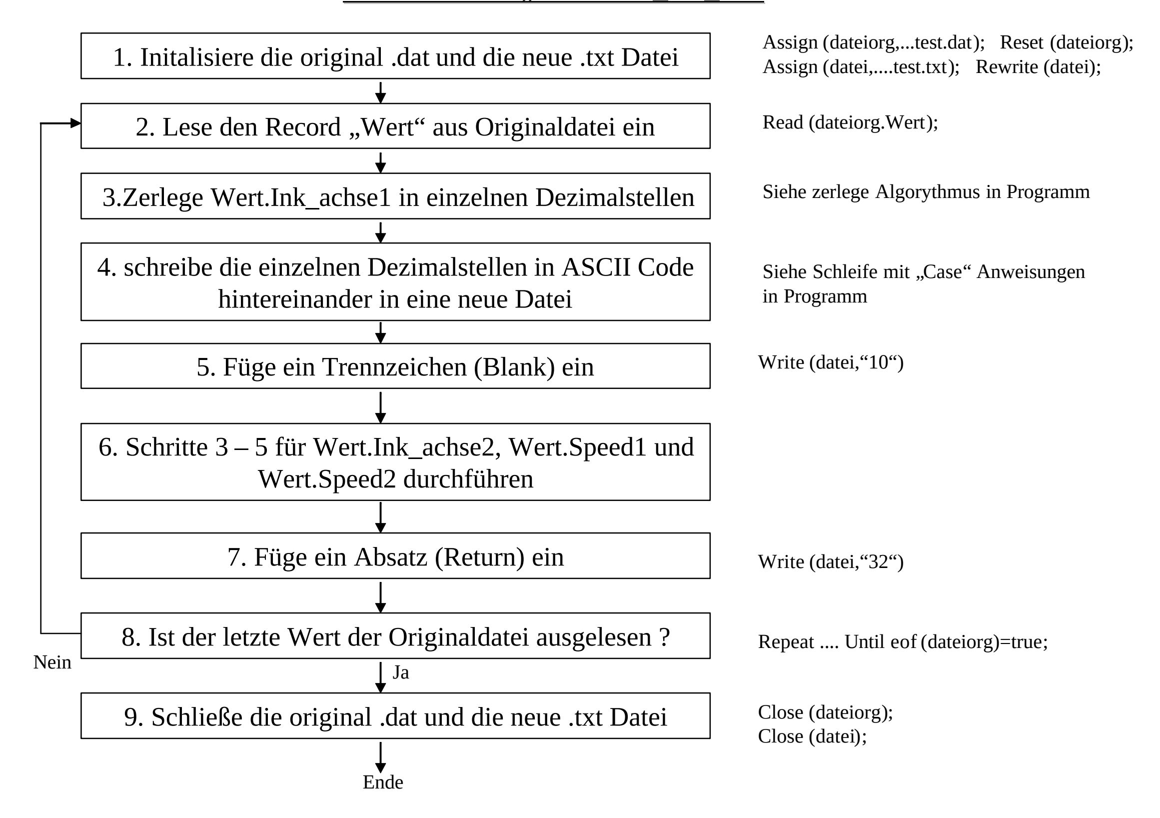
Programm „Stickall.pas“ Winkel und Winkelgeschwindigkeit anzeigen /abspeichern



Unit Prozedur „convert_dat_txt“



Unit Prozedur „convertall_dat_txt“



VII Sticksoftware Veränderungen im Detail

VII.1 Veränderungen zur Implementierung der Inkrementalgeber Software

1) Definition der neuen globalen Variablen

Die Variablen für den Wert des Ink. Gebers, für die Skalierung, den Dämpfungsfaktor und für die Geschwindigkeit werden in der Header Datei ***ik121out.h*** definiert. Dabei wird ein Verbund definiert, so dass sich die globalen Variablen *InkRoll.wert, InkRoll.scale, InkRoll.damp InkRoll.speed , InkRoll.periode; InkNick.wert, InkNick.scale, InkNick.damp, InkNick.speed, InkNickl.periode* ergeben.

Der Datentyp Ink ist dabei extern in der Datei ***ik121out..c*** definiert, in der auch die Hardware Basis I/O Adresse der Inkrementalgeberkarte definiert ist.

Weiterhin sind statische Variablen in der Datei *ik121out.c* definiert, welche für die Geschwindigkeitsberechnung in der Funktion *arrayspeed_ik121* benötigt werden.

[Auszug aus ik121out.h]
Zeile 16

```
typedef struct Ink
        {
        double wert;
        double scale;
        double damp;
        double speed;
        unsigned int periode;
        } Ink;

extern Ink InkRoll, InkNick;
extern int baseaddress;
```

[Auszug aus ik121out.c]
Zeile 19

```
Ink InkRoll, InkNick;
Int baseaddress;

static double Ink_oldvalue_roll;
static double Ink_oldvalue_nick;
static double Ink_array_rollvalue[31];   /* Ink Achse 1 = roll Achse */
static double Ink_array_nickvalue[31];   /* Ink Achse 2 = nick Achse */
static unsigned int index1=1;
static unsigned int index2=1;
```

Den Inhalt der oben definierten globalen Variablen, soll folgende Tabelle darstellen:

.wert	Inkremente des Inkrementgebers der jeweiligen Achse
.scale	Skalierfaktor, der das mechnische Übersetzungsverhältnis und die Skalierung von Inkrementen auf Grad beinhaltet.
.damp	Dämpfungswert, der die Dämpfung proportional zur Geschwindigkeit erhöht. !Wichtig! Dieser Wert darf nicht willkürlich verändert werden ,

	sondern nur mit den in Tabelle IV.3 beschriebenen Bereichen
.speed	Geschwindigkeit, welche mit der Funktion arrayspeed_ik121 berechnet wird
.periode	Zeitvorgabe für das Ableitungsintervall. Variierbar von 1-30 (entspricht 1 -30 ms)

2) Schreiben der Konstanten aus der Konfigerationsdatei stick.cfg

Da es wünschenswert ist, alle Parameter die zum Betrieb des Sticks notwendig sind in einer Datei verändern zu können, wurde die Skalierfaktoren für die Inkremantalgeber in der gleichen Datei (Stick.cfg) definiert, wie die restlichen Faktoren für den Stick.

Die Datei ***Stick.cfg*** wird mit der Funktion *print_channels* aus der Datei ***get_sys.c*** gelesen und ausgewertet. Zuerst werden die einzelnen Abschnitte („**section**") definiert und wenn der Ausdruck „Begin *section*" in Stick.cfg mit der [if Abfrage] gefunden wurde, dann wird in einem weitern Abschnitt [case Abfrage] das Umschreiben der Werte aus der Datei auf die global definierte Variablen vorgenommen.

Beispiel für Ink Geber in ***get_sys.c***:

[Generelle Definierung von „Section" als enum]

Zeile 93

```
enum { NONE, CHANNELS, CONTROLER, KOMMUT, TORQUE, MOTOR, INK, END=99} section;
```

(Bemerkung: "enum" ist eine Aufzählungstyp, um z.B. die Abfrage von case Anweisungen übersichtlicher zu gestallten)

[If Abfrage]

Zeile 145

```
if (!section && !memicmp(line, "Begin Ink", 9)))
        {
        section = INK;
        continue;
        }
```

[Case Abfrage]

Zeile 412

```
case INK:     /*    Inkrementalgeber Section */

           parnr = sscanf(line, "%s %lf", name, &scale );

           #ifdef DEBUG
           if (debug)
             {
             printf("GET CONFIG:   line = %s\n", line);
             printf("GET CONFIG:   name = %s\n", name);
             printf("GET CONFIG:   scale = %f\n", scale);
             }
           #endif

           if (parnr != 2)
             {
```

```
    ERROR_STR(E_WRONG_SYNTAX, line);
    ERROR(E_WRONG_CONFIG_FILE);
    return E_CANT_INIT;
    }

  if (!memicmp(name, "NickScale", 8))
     InkNick.scale = scale;
  else if  (!memicmp(name, "RollScale", 8))
     InkRoll.scale = scale;
  else if (!memicmp(name, "NickDamp", 8))
     InkNick.damp = scale;
  else if  (!memicmp(name, "RollDamp", 8))
     InkRoll.damp = scale;
  else if (!memicmp(name, "NickPeriode", 8))
     InkNick.periode = scale;
  else if (!memicmp(name, "RollPeriode", 8))
     InkRoll.periode = scale;
  else if (!memicmp(name, "baseaddress", 8))
     baseaddress = scale;
  else          {
    ERROR_STR(E_WRONG_TRC_MODE, name);
    ERROR(E_WRONG_CONFIG_FILE);
    return E_CANT_INIT;
    }

  break;
```

[Auszug aus der Datei Stick.cfg]

```
# Scalierungswerte für Inkrementalgeber
# Konstante baseaddress (Hex=0x330 => Dez=816)
#
Begin Ink
NickScale    0.00058877866
RollScale    0.00058877866
Nickdamp    1
Rolldamp     1
Nickperiode  30
Rollperiode   30
baseaddress  816
end
```

3) Initalisierung und Abrufen der Ink. Geber Daten in der Stick Software

Die folgende Prozeduren werden in der Stick Software aufgerufen, um eine Messwertaufnahme/Verarbeitung der Inkrementalgeber zu ermöglichen. Alle diese Prozeduren/Funktionen befinden sich in der Datei ik121out.c und ik121_0.c im „\low_lvl“ Verzeichnis und werden mit den anderen Quelldateien in diesem Verzeichnis über den Befehl „make“ und der damit verbundenen Konfigerationsdatei „makekonfig“ in die Libary ad_dma_l.lib kompiliert.

[Initalisierung]

In die Datei *stick.c* wird in der Prozedur *StickInit* in Zeile 242 die Prozedur *init_ik121* aufgerufen, um die Messkarte der Ink. Geber zu initialisieren und für den ersten Messwertabruf bereit zu stellen.

[Abruf]

In der Datei *komukomp.c* ist die Prozedur *get_data* implemtiert, welche im Hauptalgorithmus mit einem durch den RT-Kernel garantierte Periode von 1ms aufgerufen wird. Innerhalb dieser Prozedur wird die Funktion *getvalue_ik121* aufgerufen, welche bei jedem Durchlauf den aktuellen Zählerstand des Ink. Geber der jeweiligen Achse ausgibt und in die Global zur Verfügung stehende Variablen schreibt. Die weitere Geschwindigkeitsberechnung wird direkt danach durchgeführt, indem die Funktion *getarrayvalue_ik121* aufgerufen wird.

[Auszug komukomp.c]

Zeile 131

```
/* Meßwertabruf und Verarbeitung der Inkrementalgeber */

        InkRoll.wert = getvalue_ik121 (baseaddress,1)*InkRoll.scale;
        InkRoll.speed = arrayspeed_ik121 (1);
```

4) Verarbeitung der Meßwerte

Die weitere Verarbeitung der Messwerte geschieht in *stick.c* in der Prozedur *RollStick*, bzw. *NickStick*. Diese Prozeduren stellen den eigentlichen Regelkreislauf dar, welcher solange mit höchster Priorität läuft, wie der Stick aktiv ist. Innerhalb dieser Prozedure ist auch der Summationspunkt von Soll und Istwert vorhanden, auf den zur Dämpfung des Systems das Geschwindigkeitssignal (multipliziert mit dem Dämpfungsfaktor) zusätzlich aufsummiert wird. Siehe dazu das Regelungsblockschaltbild in Abschnitt I.

[Auszug Stick.c]

Zeile 140

```
/* Regler aufrufen und TRC- Werte addieren. In Strom-Inremente wandeln */
        I_StellRoll= (ReglerRoll(M_SollRoll, RollData.moment) + M_CompRoll)
                  * RollData.MotorConst + (InkRoll.speed*InkRoll.damp);
```

Zeile 207

```
/* Regler aufrufen und TRC- Werte addieren. In Strom-Inremente wandeln */
        I_StellNick = (ReglerNick(M_SollNick, NickData.moment) + M_CompNick)
                  * NickData.MotorConst – (InkNick.speed*InkNick.damp);
```

5) Bildschirmausgabe des Geschwindigkeitsignals

In der Datei *Sserve.c* ist zur Bildschirmausgabe der Regler Soll- und Istwert die Prozedur *OutpWatch* definiert. Dort kann eine zusätzliche Angabe des Geschwindigkeitssignals des Ink. Gebers (*InkRoll.speed*, *InkNick.speed*) erfolgen durch Ergänzen der Wprintf Anweisung erfolgen.

[Auszug SServe.c]

Zeile 82

```
void OutpWatch(void)  /* Task um die Ausgangsdaten anzuzeigen */
```

```
{
int i;
Window *W;

W = NewWindow(31, 0, 79, 3, 1024, 0x31, "Output");

while (True)
	{
	GotoXY(W, 0, 0);
	Wprintf(W, "Roll: %9.3f  K-M: %5.3f Ink: %10.3f",
				RollData.StellMoment, RollData.KMoment, InkRoll.speed*InkRoll.scale);
	GotoXY(W, 0, 1);
	Wprintf(W, "Nick: %9.3f  K-M: %5.3f Ink: %10.3f" ,
				NickData.StellMoment, NickData.KMoment, InkNick.speed*InkNick.scale);

	RTKDelay(BS_DELAYTIME);
	}
}
```

VII.2 Veränderungen der Sticksoftware zur Kopplung der Sticks

1) Definition der neuen globalen Variablen

```
typedef struct Koppel
        {
        double wert;
        double offset;
        double scale;
        double steigung;
        int kennlinie;
        } Koppel;

extern Koppel Rollkoppel,Nickkoppel;
```

.wert	Aktueller Wert des Kopplungssignals
.offset	Offsetwert des Kopplungssignals bei Sticknullstellung. Wird in der Software von .wert abgezogen.
.scale	Skalierfaktor zur Anpassung der Gewichtung des Kopplungssignals an die restlichen Regelungsgrößen
.steigung	Steigung der Kennlinie bei entsprechender Kennlinienauswahl
.kennlinie	Gibt die gewünschten Kennlinie zur Kopplung an

2) Schreiben der Konstanten aus der Konfigerationsdatei stick.cfg

Genau wie auch schon die Konstanten für die Inkrementalgeber Implementierung, werden die Konstanten für die Kopplung aus der Datei Stick.cfg in der Datei in der Procedur *print_channels (void)* (in Get_sys.c) ausgewertet und als globale Variablen zur Verfügung gestellt.
Zuerst wird noch ein weiterer Abschnitt („**section**") KOPPEL definiert und wenn durch den
Ausdruck „Begin *KOPPEL*" in Stick.cfg gefunden wird [if Abfrage], dann wird in einem weiteren Abschnitt [case Abfrage] das Umschreiben der Werte aus der Datei auf Global definierte Variablen vorgenommen.

[Generelle Definierung von „Section" als enum]
Zeile 93

```
enum { NONE, CHANNELS, CONTROLER, KOMMUT, TORQUE, MOTOR, INK, KOPPEL,END=99} section;
```

[If Abfrage]
Zeile 145

```
if (!section && !memicmp(line, "Begin Koppel", 12))
                        {
                        section = KOPPEL;
                        continue;
                        }
```

[Case Abfrage]
Zeile 455

```
case KOPPEL:    /*  Koppel Section */

        parnr = sscanf(line, "%s %lf", name, &scale );
```

```
#ifdef DEBUG
if (debug)
  {
  printf("GET CONFIG:   line = %s\n", line);
  printf("GET CONFIG:   name = %s\n", name);
  printf("GET CONFIG:   scale = %f\n", scale);
  }
#endif
if (parnr != 2)
  {
  ERROR_STR(E_WRONG_SYNTAX, line);
  ERROR(E_WRONG_CONFIG_FILE);
  return E_CANT_INIT;
  }
if  (!memicmp(name, "RollOffset", 10))
        Rollkoppel.offset = scale;
else if  (!memicmp(name, "NickOffset", 10))
        Nickkoppel.offset = scale;
else if (!memicmp(name, "RollScale", 9))
        Rollkoppel.scale = scale;
else if  (!memicmp(name, "NickScale", 9))
        Nickkoppel.scale = scale;
else if  (!memicmp(name, "RollKLsteigung", 14))
        Rollkoppel.steigung = scale;
else if  (!memicmp(name, "NickKLsteigung", 14))
        Nickkoppel.steigung = scale;
else if  (!memicmp(name, "RollKennlinie", 13))
        Rollkoppel.kennlinie = scale;
else if  (!memicmp(name, "NickKennlinie", 13))
        Nickkoppel.kennlinie = scale;
else
{
ERROR_STR(E_WRONG_TRC_MODE, name);
ERROR(E_WRONG_CONFIG_FILE);
return E_CANT_INIT;
}
```

[Auszug aus der Datei Stick.cfg]

```
# Scallierungswerte für Kopplung
#
Begin Koppel
RollOffset        0
NickOffset        0
RollScale         1
NickScale         1
RollKLsteigung  1
NickKLsteigung  1
RollKennlinie     1
NickKennlinie     1
End.
```

3) Realisierung der Kopplung

Wie in dem Regelungsblockschaltbild in Abschnitt I dargestellt, wird das Kopplungssignal an der gleichen Stelle wie das Inkrementalgeber Geschwindigkeitssignal aufsummiert. Dazu wird in der Prozedur Rollstick bzw. Nickstick in der Datei Stick.c am Regelungs Summationspunkt der Wert der Funktion Koppelroll bzw. Koppelnick aufsummiert.

[Auszug Stick.c]
Zeile 142

```
/* Regler aufrufen und TRC- Werte addieren. In Strom-Inremente wandeln */
```

```
        I_StellRoll= (ReglerRoll(M_SollRoll, RollData.moment) + M_CompRoll) *
                    RollData.MotorConst
                     + (InkRoll.speed*InkRoll.damp)              /* zuzätzliche Dämpfung */
                     + Koppelroll();                            /* Kopplung mit CoStick */
```

Zeile 208

```
/* Regler aufrufen und TRC- Werte addieren. In Strom-Inremente wandeln */
        I_StellNick = (ReglerNick(M_SollNick, NickData.moment) + M_CompNick)
                     * NickData.MotorConst
                      - (InkNick.speed*InkNick.damp)         /* zuzätzliche Dämpfung */
                     + Koppelnick ();                       /* Kopplung mit CoStick */
```

4) Bildschirmausgabe des Koppelsignals

Um zu Überprüfungszwecken das Koppelsignal auf dem Bildschirm darzustellen, wurde in der Datei *Sserve.c* die Bildschirmausgabe der NickData.passiv und RollData.passiv Variablen realisiert. Dazu wurde in der Prozedure InpWatch die Window Ausgabe verändert:

[Auszug SServe.c]

Zeile 62

```
void InpWatch(void)  /* Task um die Eingangsdaten anzuzeigen */
        {
        int i;
        Window *W;

        W = NewWindow(0, 0, 30, 3, 256, 0x31, "Input");

        while (True)
                {
                GotoXY(W, 0, 0);
                Wprintf(W, "R: P %6.1lf M %6.1lf", RollData.phi, RollData.passiv);
                GotoXY(W, 0, 1);
                Wprintf(W, "N: P %6.1lf M %6.1lf", NickData.phi, NickData.passiv);

                RTKDelay(BS_DELAYTIME);
                }
        }
```

VIII Softwarecode Testprogramme

Unit Stickall.pas

```
program stickall;

(* Programm zur Aufnahme der Strecke und der Geschwindigkeit eines Ink.Gebers *)

USES crt,ik121_0,converta,control;

CONST
  base_address = $330;
  time = 0;
  k = 360/1048*1714/1000;

Type datensatz=record       Ink_achse1,Ink_achse2,Speed1,Speed2:comp;
                          end;

VAR              Wert:datensatz;
             dateiorg:file of datensatz;
      Wertalt1,Wertalt2:comp;

BEGIN
clrscr;
  (* initalisierung der Dateien: "C:\tests\test1.dat" *)
assign (dateiorg,'c:\tests\test1.dat');
rewrite (dateiorg);


  (* initalisierung board im interpolations Mode, axis 1 und axis 2 *)
write_g26 (base_address, 1, $0c, $0045);
write_g26 (base_address, 2, $0c, $0045);

  (* Reset Error bit, start counter, axis 1 und axis 2 *)
write_g26 (base_address, 1, $0e, $0068);
write_g26 (base_address, 2, $0e, $0068);

  (* Write to control register 2, axis 1 und axis 2 *)
write_g26 (base_address, 1, $1c, $0028);
write_g26 (base_address, 2, $1c, $0028);

                   (* Beide Achsen werden gemessen *)

(* Erst Meáwertaufnahme *)

  (* Software latch in register 0, axis 1 und axis 2 *)
write_g26 (base_address,1,$e,$1);
write_g26 (base_address,2,$e,$1);

  (* Poll whether latched in axis 1 und achse 2*)
poll_latch (base_address, 1, 0);
poll_latch (base_address, 2, 0);

  (* Read axis 1 , und Normierung *)
Wert.Ink_achse1:= read_count_value48 (base_address, 1, 0,false)*k;
Wert.Ink_achse2:= read_count_value48 (base_address, 2, 0,false)*k;

Wertalt1:=Wert.ink_achse1;
Wertalt2:=Wert.ink_achse2;

(* Zweite Messwertaufnahme *)

Repeat
```

```
  (* Software latch in register 0, axis 1 und axis 2 *)
write_g26 (base_address,1,$e,$1);
write_g26 (base_address,2,$e,$1);

  (* Poll whether latched in axis 1 und achse 2*)
poll_latch (base_address, 1, 0);
poll_latch (base_address, 2, 0);

  (* Read axis 1 , und Normierung *)
Wert.Ink_achse1:= read_count_value48 (base_address, 1, 0,false)*k;
Wert.Ink_achse2:= read_count_value48 (base_address, 2, 0,false)*k;

Wert.Speed1:=Wertalt1-Wert.Ink_achse1;
Wert.Speed2:=Wertalt2-Wert.Ink_achse2;

                              (* Ausgabe *)
  gotoxy (1,1);
  writeln('Achse 1 Winkel  : ',Wert.Ink_achse1:10:0);
  writeln('Achse 1 Speed   : ',Wert.Speed1:10:0);
  writeln;
  writeln('Achse 2 Winkel  : ',Wert.Ink_achse2:10:0);
  writeln('Achse 2 Speed   : ',Wert.Speed2:10:0);

  Wertalt1:=Wert.Ink_achse1;   (* F•  r neue Speed berechnung *)
  Wertalt2:=Wert.Ink_achse2;

  (* Ausdruck der Position und der Geschwindigkeiten in eine Datei *)

  write (dateiorg,Wert);

  (* Ausdruck der Datenregister auf den Bildschirm   *)
 (* gotoxy (5,7); *)
 (* outputreg (base_address,1); *)

delay (time);

UNTIL KEYPRESSED;

close (dateiorg);

convertall_dat_txt;
END.
```

Unit converta.pas

```
Unit converta;  (* designed by Daniel Prutz *)

(* Dieses programm liest Zahlen aus einer Datei aus, um sie wieder in eine
  neue Datei abzuspeichern, aber so, daá sie als zahlen in der neuen Datei
  direkt lesbar und z.B. mit Exel auswertbar sind.
  Dieses Problem entsteht bei Pascal dadurch, daá alle Zeichen die in eine
  Datei normal abgespeicher werden in ASCII Format umgewandelt werden.
  Normale Zahlen werden daher ebenfalls in ASCII umbewandelt und sind in
  der Datei selber nicht zu entschl•  sseln *)

Interface

Procedure convertall_dat_txt;

Implementation

uses crt;

Procedure convertall_dat_txt;
```

```
const  Trennzeichen=10;        (* Werte f•  r ASCII Zeichen *)
       Minus=45;
       Leerzeichen=32;

Type datensatz=record     Ink_achse1,Ink_achse2,Speed1,Speed2:comp; end;


Var                  Wert:datensatz;
                dateiorg:file of datensatz;
               WertASCII:byte;
                dateineu:file of byte;
                  stelle:array[1..6] of longint;
                     i:byte;
    Wert_achse1,Wert_achse2:longint;


Begin
  assign (dateineu,'a:\test1.txt');
  rewrite (dateineu);
  assign (dateiorg,'C:\tests\test1.dat');
  reset (dateiorg);
  repeat
   read (dateiorg,Wert);

   (* Wert auf integer Niveau bringen (Nachkommastellen aschneiden)
   Vorsicht !!: bei Normierung drauf auchten, das Nachkommastellen keine
   Bedeutung haben !!
   Wenn darauf geachtet wird, existiert kein Genauigkeitsverluát *)

   Wert_achse1:=trunc(Wert.Ink_achse1);

   (* Vorzeichen dedektieren und vor Zahl schreiben *)
   (* Wert in Betrag umsetzten *)

   If Wert_achse1<0 then
                  Begin
                   WertASCII:=Minus;
                   Wert_achse1:=Wert_achse1*(-1);
                  end
                else WertASCII:=Leerzeichen;
   Write (dateineu,WertASCII);

    (* Zerteilen der Dezimalen Zahl in einzelne Stellen f•  r Achse 1 *)

    stelle[6]:=wert_achse1 div 100000;
    wert_achse1:=wert_achse1-stelle[6]*100000;
    stelle[5]:=wert_achse1 div 10000;
    wert_achse1:=wert_achse1-stelle[5]*10000;
    stelle[4]:=wert_achse1 div 1000;
    wert_achse1:=wert_achse1-stelle[4]*1000;
    stelle[3]:=wert_achse1 div 100;
    wert_achse1:=wert_achse1-stelle[3]*100;
    stelle[2]:=wert_achse1 div 10;
    wert_achse1:=wert_achse1-stelle[2]*10;
    stelle[1]:=wert_achse1 div 1;

  (* Umrechnen des Achsen-Wertes in ASCII Format und schreiben in Datei *)

  i:=6;
    Repeat
      case stelle[i] of 0: WertASCII:=48; end;
      case stelle[i] of 1: WertASCII:=49; end;
      case stelle[i] of 2: WertASCII:=50; end;
      case stelle[i] of 3: WertASCII:=51; end;
      case stelle[i] of 4: WertASCII:=52; end;
```

```
    case stelle[i] of 5: WertASCII:=53; end;
    case stelle[i] of 6: WertASCII:=54; end;
    case stelle[i] of 7: WertASCII:=55; end;
    case stelle[i] of 8: WertASCII:=56; end;
    case stelle[i] of 9: WertASCII:=57; end;
    write (dateineu,WertASCII);
    i:=i-1;
  until i=0;
 WertASCII:=Leerzeichen;          (* Zur Trennung der Werte*)
 write (dateineu,WertASCII);

  (* Wert auf integer Niveau bringen (Nachkommastellen abrunden)
  Vorsicht !!: bei Normierung drauf auchten, das Nachkommastellen keine
  Bedeutung haben !!
  Wenn darauf geachtet wird, existiert kein Genauigkeitsverluát *)

  Wert_achse2:=trunc(Wert.Ink_achse2);

(* Vorzeichen dedektieren und vor Zahl schreiben *)
(* Wert in Betrag umsetzten *)

  If Wert_achse2<0 then
                Begin
                  WertASCII:=minus;
                  Wert_achse2:=Wert_achse2*(-1);
                end
              else WertASCII:=Leerzeichen;
  Write (dateineu,WertASCII);

  (* Zerteilen der Dezimalen Zahl in einzelne Stellen f• r Achse 2 *)

 stelle[6]:=wert_achse2 div 100000;
 wert_achse2:=wert_achse2-(stelle[6]*100000);
 stelle[5]:=wert_achse2 div 10000;
 wert_achse2:=wert_achse2-(stelle[5]*10000);
 stelle[4]:=wert_achse2 div 1000;
 wert_achse2:=wert_achse2-(stelle[4]*1000);
 stelle[3]:=wert_achse2 div 100;
 wert_achse2:=wert_achse2-(stelle[3]*100);
 stelle[2]:=wert_achse2 div 10;
 wert_achse2:=wert_achse2-(stelle[2]*10);
 stelle[1]:=wert_achse2 div 1;

 (* Umrechnen des Achsen-Wertes in ASCII Format und schreiben in Datei *)


 i:=6;
  Repeat
    case stelle[i] of 0: WertASCII:=48; end;
    case stelle[i] of 1: WertASCII:=49; end;
    case stelle[i] of 2: WertASCII:=50; end;
    case stelle[i] of 3: WertASCII:=51; end;
    case stelle[i] of 4: WertASCII:=52; end;
    case stelle[i] of 5: WertASCII:=53; end;
    case stelle[i] of 6: WertASCII:=54; end;
    case stelle[i] of 7: WertASCII:=55; end;
    case stelle[i] of 8: WertASCII:=56; end;
    case stelle[i] of 9: WertASCII:=57; end;
    write (dateineu,WertASCII);
    i:=i-1;
  until i=0;
 WertASCII:=Leerzeichen;          (* Zur Trennung der Werte*)
 write (dateineu,WertASCII);

     (**** Bearbeiten der Geschwindigkeitsdaten ******)
```

```
(* Vorsicht !!: bei Normierung drauf auchten, das Nachkommastellen keine
   Bedeutung haben !!
   Wenn darauf geachtet wird, existiert kein Genauigkeitsverluát *)

   Wert_achse1:=trunc(Wert.Speed1);

   (* Vorzeichen dedektieren und vor Zahl schreiben *)
   (* Wert in Betrag umsetzten *)

   If Wert_achse1<0 then
                  Begin
                   WertASCII:=Minus;
                   Wert_achse1:=Wert_achse1*(-1);
                  end
                else WertASCII:=Leerzeichen;
   Write (dateineu,WertASCII);

    (* Zerteilen der Dezimalen Zahl in einzelne Stellen f•  r Achse 1 *)

    stelle[6]:=wert_achse1 div 100000;
    wert_achse1:=wert_achse1-stelle[6]*100000;
    stelle[5]:=wert_achse1 div 10000;
    wert_achse1:=wert_achse1-stelle[5]*10000;
    stelle[4]:=wert_achse1 div 1000;
    wert_achse1:=wert_achse1-stelle[4]*1000;
    stelle[3]:=wert_achse1 div 100;
    wert_achse1:=wert_achse1-stelle[3]*100;
    stelle[2]:=wert_achse1 div 10;
    wert_achse1:=wert_achse1-stelle[2]*10;
    stelle[1]:=wert_achse1 div 1;

  (* Umrechnen des Achsen-Wertes in ASCII Format und schreiben in Datei *)

  i:=6;
    Repeat
      case stelle[i] of 0: WertASCII:=48; end;
      case stelle[i] of 1: WertASCII:=49; end;
      case stelle[i] of 2: WertASCII:=50; end;
      case stelle[i] of 3: WertASCII:=51; end;
      case stelle[i] of 4: WertASCII:=52; end;
      case stelle[i] of 5: WertASCII:=53; end;
      case stelle[i] of 6: WertASCII:=54; end;
      case stelle[i] of 7: WertASCII:=55; end;
      case stelle[i] of 8: WertASCII:=56; end;
      case stelle[i] of 9: WertASCII:=57; end;
      write (dateineu,WertASCII);
      i:=i-1;
    until i=0;
  WertASCII:=Leerzeichen;          (* Zur Trennung der Werte*)
  write (dateineu,WertASCII);

   (* Wert auf integer Niveau bringen (Nachkommastellen abrunden)
   Vorsicht !!: bei Normierung drauf auchten, das Nachkommastellen keine
   Bedeutung haben !!
   Wenn darauf geachtet wird, existiert kein Genauigkeitsverluát *)

   Wert_achse2:=trunc(Wert.Speed2);

 (* Vorzeichen dedektieren und vor Zahl schreiben *)
 (* Wert in Betrag umsetzten *)

   If Wert_achse2<0 then
                  Begin
                    WertASCII:=minus;
                    Wert_achse2:=Wert_achse2*(-1);
                  end
```

```
              else WertASCII:=Leerzeichen;
   Write (dateineu,WertASCII);

   (* Zerteilen der Dezimalen Zahl in einzelne Stellen f• r Achse 2 *)

  stelle[6]:=wert_achse2 div 100000;
  wert_achse2:=wert_achse2-(stelle[6]*100000);
  stelle[5]:=wert_achse2 div 10000;
  wert_achse2:=wert_achse2-(stelle[5]*10000);
  stelle[4]:=wert_achse2 div 1000;
  wert_achse2:=wert_achse2-(stelle[4]*1000);
  stelle[3]:=wert_achse2 div 100;
  wert_achse2:=wert_achse2-(stelle[3]*100);
  stelle[2]:=wert_achse2 div 10;
  wert_achse2:=wert_achse2-(stelle[2]*10);
  stelle[1]:=wert_achse2 div 1;

  (* Umrechnen des Achsen-Wertes in ASCII Format und schreiben in Datei *)


  i:=6;
    Repeat
      case stelle[i] of 0: WertASCII:=48; end;
      case stelle[i] of 1: WertASCII:=49; end;
      case stelle[i] of 2: WertASCII:=50; end;
      case stelle[i] of 3: WertASCII:=51; end;
      case stelle[i] of 4: WertASCII:=52; end;
      case stelle[i] of 5: WertASCII:=53; end;
      case stelle[i] of 6: WertASCII:=54; end;
      case stelle[i] of 7: WertASCII:=55; end;
      case stelle[i] of 8: WertASCII:=56; end;
      case stelle[i] of 9: WertASCII:=57; end;
      write (dateineu,WertASCII);
      i:=i-1;
    until i=0;
  WertASCII:=Trennzeichen;          (* Zur Trennung der Werte*)
  write (dateineu,WertASCII);
until eof (dateiorg)=true;
close (dateineu);
close (dateiorg);
end;
end.
```

Unit Control.pas

```
unit control;  (* unit designed by Daniel Prutz *)

Interface

uses crt,ik121_0;

Procedure converthb (datum:word);
Procedure Outputreg (base_adress:word;axes:byte);


Implementation

(* Diese Procedure Convertiert die Hexadizimale Zahl in eine Bin„re Zahl
   damit die einzelnen Bits im Register direkt abgelesen werden k”nnen
   Die Bin„re Zahl ist bei der Darstellung genau vertauscht (links=Low Bit) *)


Procedure converthb (datum:word);
```

```
Const n=16;

Var x:array[1..n] of byte;
    i:integer;

Begin
    For i:=1 to n do
    Begin
      x[i]:=datum mod 2;
      datum:=datum div 2;
    end;
  i:=n;
  While i>0 do
  Begin
    Write (x[i]);
    If i=(n/2+1) then Write (' ');
    i:=i-1;
  end;
end;


(* Dieses Procedure zeigt alle Lesbaren Register einer Achse an *)

Procedure outputreg (base_adress:word;axes:byte);

Var Wert:word;

Begin
    writeln (' Lesezugriffe Register ');
    writeln ('                                    Bit : FEDCBA98 76543210');
    write ('Daten Register 0, LS-Word              : ');
    converthb (read_g26 (base_adress,axes,$00));
    writeln;
    write ('Daten Register 0                       : ');
    converthb (read_g26 (base_adress,axes,$02));
    writeln;
    write ('Daten Register 0, WS-Word              : ');
    converthb (read_g26 (base_adress,axes,$04));
    writeln;
    write ('Daten Register 1, LS-Word              : ');
    converthb (read_g26 (base_adress,axes,$06));
    writeln;
    write ('Daten Register 1                       : ');
    converthb (read_g26 (base_adress,axes,$08));
    writeln;
    write ('Daten Register 1, WS-Word              : ');
    converthb (read_g26 (base_adress,axes,$0A));
    writeln;
    Writeln;
    write ('Initalisierungsregister Register  "0Ch" : ');
    converthb (read_g26 (base_adress,axes,$0C));
    writeln;
    write ('Status Register                   "0Eh" : ');
    converthb (read_g26 (base_adress,axes,$0E));
    Writeln;
    write ('Amplitudenwertregister  Register  "10h" : ');
    converthb (read_g26 (base_adress,axes,$10));
    writeln;
    write ('Interrupt Status Register         "14h" : ');
    converthb (read_g26 (base_adress,axes,$14));
    writeln;
    write ('Amplitude f•  r 0ø Signal         "16h" : ');
    converthb (read_g26 (base_adress,axes,$16));
    writeln;
    write ('Amplitude f•  r 90ø Signal        "18h" : ');
    converthb (read_g26 (base_adress,axes,$18));
```

```
    writeln;
    write ('Status 3/Kennungs Register        "1Ch" : ');
    converthb (read_g26 (base_adress,axes,$1C));
    writeln;
    write ('Status Register 4                 "1Eh" : ');
    converthb (read_g26 (base_adress,axes,$1E));
    writeln;
end;
end.
```

IX Softwarecode Inkrementalgeber Implementierung

ik121out.c

```
/*---------------------Ik121out.C---------------------

  Dieses Programm stellt die erweiterten Funktionen zur direketen
  Messwerterfassung der Inkrementalgeber zur verfügung.


 V 1.7
 Dez 2001

 Project files:      IK121_0.C, IK121out.C
 Include files:       IK121_0.H, IK121out.H
 ----------------------------------------------------*/

#include <stdio.h>
#include <conio.h>
#include "ik121_0.h"
#include "ik121out.h"

Ink InkRoll,InkNick;
int baseaddress;

static double Ink_oldvalue_roll;
static double Ink_oldvalue_nick;
static double Ink_array_rollvalue[31];   /* Ink Achse 1 = roll Achse */
static double Ink_array_nickvalue[31];   /* Ink Achse 2 = nick Achse */
static unsigned int index1=1;
static unsigned int index2=1;

void init_ik121 (int base_address)
{
             /* Initialise the board in interpolation mode, axis 1 and axis 2*/
write_g26 (base_address, 1, 0x0c, 0x0045);
write_g26 (base_address, 2, 0x0c, 0x0045);

             /* Reset error bit, reset counter, start counter axis 1 and axis 2*/
write_g26 (base_address, 1, 0x0e, 0x0068);
write_g26 (base_address, 2, 0x0e, 0x0068);

             /* Write to control register 2, axis 1 and axis 2*/
write_g26 (base_address, 1, 0x1c, 0x0028);
write_g26 (base_address, 2, 0x1c, 0x0028);
}

double getvalue_ik121 (int base_address,int axes)
{
  double Value;
  if (axes == 1)
  {
             /* Software latch in register 0 (axes 1)*/
  soft_l0 (base_address, axes);
             /* Poll whether latched */
  poll_latch (base_address, axes, 0);
             /*read 48 bit value from Data register*/
  Value = read_count_value48 (base_address, axes, 0);
  }
  else if (axes == 2)
  {
```

```
 /* Software latch in register 1 (axis 2) */
 soft_l1 (base_address,axes);
 /* Poll whether latched */
 poll_latch (base_address, axes, 1);
 /* read 48 bit value from Data register */
 Value = read_count_value48 (base_address, axes, 1);
 }
 else printf ("Fehler bei Achsangabe (Printet by getvalue");
 return Value;
}

double arrayspeed_ik121 (int axes)
{
 unsigned int i;
 double summe;

 if (axes == 1)    /* Roll Achse*/
 {
        if (index1 < InkRoll.periode) index1 = index1 + 1;
        else index1 = 0;
        Ink_array_rollvalue[index1] = Ink_oldvalue_roll - InkRoll.wert;
        Ink_oldvalue_roll = InkRoll.wert;
        summe = 0;
        for (i=0;i<=(InkRoll.periode);i++) summe = summe + Ink_array_rollvalue[i];
                summe = summe/InkRoll.periode;
 }

 else if (axes == 2)   /* Nick Achse */
 {
        if (index2 < InkNick.periode) index2 = index2 + 1;
        else index2 = 1;
        Ink_array_nickvalue[index2] = Ink_oldvalue_nick - InkNick.wert;
        Ink_oldvalue_nick = InkNick.wert;
        summe = 0;
        for (i=0;i<=InkNick.periode;i++) summe = summe + Ink_array_nickvalue[i];
        summe = summe/InkNick.periode;
 }
 else
         {
         printf("Fehler bei Achsangabe (Printet by Function arrayspeed)");
         summe = 0;
         }
 return summe;
}


double inkspeed (int axes)
{
 double value;
 extern double Ink_oldvalue_roll;
 extern double Ink_oldvalue_nick;

 if (axes == 1)
        {
         value = Ink_oldvalue_roll - InkRoll.wert;
         Ink_oldvalue_roll = InkRoll.wert;
        }
        else if (axes == 2)
         {
          value = Ink_oldvalue_nick - InkNick.wert;
          Ink_oldvalue_nick = InkNick.wert;
         }
         else
          {
          printf("Fehler bei Achsangabe (Printet by Function Inkspeed)");
          value = 0;
```

```
                }
  return (value);
}

double timetest (int index)
{
  extern unsigned int index1;
  extern unsigned int index2;

  if (index == 1)
              {
              index1=index1 + 1;
              return index1;
              }
  else if (index == 2)
              {
              index2=index2 + 1;
              return index2;
              }
              else
                {
                printf("Fehler bei Achsangabe (Printet by Function Timetest)");
                return (0);
                }
}
```

ik121out.h

```
V 1.7
 Dez 2001

 -------------------------------------------------------*/
void init_ik121 (int base_address);
double getvalue_ik121 (int axes,int base_address);
double arrayspeed_ik121 (int axes);
double inkspeed (int axes);
double timetest (int index);

typedef struct
  {
               double wert;
               double scale;
               double damp;
               double speed;
               unsigned int periode;
               } Ink;

extern Ink InkRoll,InkNick;
extern int baseaddress;
```

Koppel.c

```
/* Diese Funktionen sind für die Drehmomentkoppelung der Sticks nötig
V 1.0 Januar 2002  Daniel Prutz
 */


#include <math.h>
#include <stdio.h>
#include <koppel.h>
#include <stick.h>

Koppel Nickkoppel, Rollkoppel;

double Koppelnick ()
  {
        Nickkoppel.wert = (NickData.passiv - Nickkoppel.offset);
  if (Nickkoppel.kennlinie = 1)
                return (Nickkoppel.wert*Nickkoppel.scale);
        else if (Nickkoppel.kennlinie = 2)
                return ((Nickkoppel.wert*Nickkoppel.steigung)*Nickkoppel.scale);
        else return (0);
        }

double Koppelroll ()
        {
        Rollkoppel.wert = (RollData.passiv-Rollkoppel.offset);

        if (Rollkoppel.kennlinie = 1)
                return (Rollkoppel.wert*Rollkoppel.scale);
        else if (Rollkoppel.kennlinie = 2)
                return ((Rollkoppel.wert*Rollkoppel.steigung)*Rollkoppel.scale);
        else return (0);
        }
```

Koppel.h

```
/* Diese header Datei ist für die kopplungs Programmierung
V 1.0       18.01.02
(Daniel Prutz Jan 2002) */


double Koppelnick (void);
double Koppelroll (void);

typedef struct Koppel
        {
        double wert;
        double offset;
        double scale;
        double steigung;
        int kennlinie;
        } Koppel;

extern Koppel Rollkoppel,Nickkoppel;
```